Trabajo Fin de Grado

2016-2017

“GARDEN-IT”

DISEÑO Y DESARROLLO DE UN SISTEMA DOMÓTICO PARA LA GESTIÓN DEL RIEGO

Autor: Alejandro Ortega Ibáñez

Tutor: Álvaro Luis Bustamante

Agradecimientos

Quería agradecer en primer lugar a mi familia el apoyo ofrecido a lo largo de toda mi etapa educativa, en especial a mi padre, Félix Ortega, y mi madre, Gema Ibáñez. Destacar y valorar el esfuerzo económico que han realizado para poder otorgarnos, tanto a mi hermano como a mí, la mejor formación posible. Gracias.

En segundo lugar, agradecer a los profesores y otros integrantes de la Universidad Carlos III de Madrid, que hacen posible una educación pública de alta calidad. Especial mención para Álvaro Luis Bustamante, tutor de mi proyecto, agradecerle su apoyo, ayuda y orientación a lo largo del desarrollo del TFG. Gracias.

Por último, quiero agradecer a Carmen Menéndez, su apoyo incondicional en todos los aspectos de mi vida, su ayuda, sus consejos, su compañía, su amistad y su amor. Mi compañera de todo, que me aguanta a pesar de su poca paciencia y forma parte de todas mis alegrías, presentes y futuras. Gracias Love.

Resumen

Diseño, desarrollo e implementación de un sistema de riego inteligente mediante un microcontrolador, sensores medioambientales y comunicación ZigBee y wifi, que permite el control manual o automático a través de internet, de la electroválvula integrada en el sistema. Cuenta con numerosos sensores que recogen información ambiental, para que podamos gestionar con mayor conocimiento, calidad y ahorro nuestro jardín.

Diseño de una interfaz gráfica en la plataforma Thinger.io, con la que el usuario pueda, monitorizar los datos almacenados que son recogidos por los sensores, configurar parámetros de control específicos, y controlar la electroválvula y modos del sistema de riego.

Índice

Abreviaturas

CPU: Unidad Central de Procesamiento

PWM: Modulación de Ancho de Pulso

USB: Universal Serial Bus

ICSP: Programación Serial en Circuito

UART: Sistema Universal Asíncrono de Recepción

IoT: Internet of Things

LDR: Resistencia Dependiente de la Luz

DFN: Trazadores de pistas

GND: Toma de Tierra

WPAN: Redes Inalámbricas de Área Personal

ISM: Industrial, Científico y Médico

MAH: Miliamperios/hora

AH: Amperios/hora

Xbee: Módulos de Comunicación por Radio

NiCad: Batería de Níquel-Cadmio

Vin: Entrada de voltaje a la placa del microcontrolador cuando se está usando una fuente externa de alimentación

GUI: Interfaz de Usuario Gráfica

MVP: Producto Mínimo Viable

Índice de Tablas

Índice de Figuras

1. Introducción

1.1. Contexto

En la actualidad, el trabajo y la vida diaria abarcan mucho de nuestro tiempo y con las nuevas tecnologías, dispositivos móviles y comunicaciones muchas de las tareas se han podido acortar en tiempo y hacerse de manera más eficiente.

Existen numerosos jardines tanto privados como públicos que requieren de cuidados y mantenimiento. Este proyecto persigue instalar un componente Hardware que monitorice y comunique el estado de un jardín determinado a partir de un circuito de controladores y sensores y un componente Software, por el cual podamos controlar dichos parámetros y ejecutar órdenes para realizar aperturas o cierres de válvula y, de esta manera, regar nuestro jardín si fuera necesario.

En esta aplicación el usuario tendrá acceso de una manera sencilla y eficaz a su jardín domotizado, teniendo control absoluto del sistema de regadío y dispondrá de la información recopilada por los sensores, pudiendo configurar el riego y otras funcionalidades de la forma que el usuario considere.

1.2. Motivación

A día de hoy, la tecnología está al alcance de todos, y nos permite crear herramientas y servicios para solucionar problemas y poder así invertir el tiempo en otras tareas.

La razón principal de desarrollo del proyecto es la mejora de la calidad de vida del usuario, y que pueda realizar la gestión de sus jardines de forma cómoda y rápida, invirtiendo así el tiempo en otras tareas. Además, se destaca la gran importancia de la tecnología móvil y la facilidad de uso, que hace que muchos de nosotros pasemos cada día cerca de alguno de estos dispositivos, ya sea para temas personales, de trabajo, comunicación u otros servicios.

1.3. Objetivos

El objetivo de este proyecto es crear un componente Hardware para ser instalado de manera sencilla en los jardines, que monitorice los parámetros del lugar y de esta manera el usuario desde la aplicación con el Software correspondiente pueda administrar y controlar de manera sencilla y eficaz el estado de su jardín.

Para ello se emplean dos partes:

- Hardware:
    - Crear un Hardware funcional compuesto de sensores de temperatura, humedad y controladores.
    - Comunicación en los controladores para transferir la información recogida por los sensores y recibir la información del exterior.
    - Inteligencia en los controladores para poder interpretar y ejecutar las ordenes de abertura y cierre de las válvulas electromagnéticas encargadas del paso del agua.

- Software:
    - Gestionar la información: El usuario podrá consultar el estado del lugar recibiendo la información por los sensores de temperatura y humedad.
    - Monitorización y registro de los datos: El usuario podrá comprobar el historial de los datos y ver la evolución gráficamente.
    - Ejecutar ordenes: El usuario podrá ejecutar ordenes de apertura o cierre de las válvulas del jardín para proceder al riego del mismo.
    - Programar riego: El usuario podrá programar el riego a una hora y fecha determinada.
    - Automatización del riego: El usuario podrá establecer un modo automático de riego bajo unas condiciones críticas configuradas.

2. Estado del arte

El objetivo de esta sección es establecer una relación de ventajas e inconvenientes de las diferentes tecnologías con las que se trabajará durante el proyecto, con el fin de establecer cuáles son óptimas para nuestro objetivo. Además, se realizará un estudio del mercado para verificar la viabilidad del proyecto.

2.1. Situación inicial de partida del proyecto

Cuando hablamos de jardinería, el cuidado y mantenimiento necesario de manera habitual hace que la constancia sea cargante, teniendo que realizar desplazamientos a las zonas indicadas y proceder al riego del lugar. Con la evolución de la tecnología y los dispositivos móviles, muchas de estas tareas y otras de semejantes características han sido optimizadas (1).

Cada vez son más las personas que utilizan la tecnología y dispositivos móviles, y se aprovechan de los servicios que estas ofrecen (1).

En 2016 realizó un informe sobre la conectividad y acceso a la información de la sociedad, en él queda reflejada la rápida digitalización de nuestra sociedad.

El 78,7% de la población española se conecta regularmente a internet, siendo el teléfono móvil el principal dispositivo a través del que se conectan (88,3%) mientras que ordenadores y portátiles se sitúan en la segunda posición (78,2%) (1).

2.2. Análisis de proyectos existentes en el mercado

En esta sección se analizarán alguno de los proyectos presentes en el mercado cuyo propósito es similar al objetivo que perseguimos y de esta manera analizar buenas y malas prácticas y obtener conclusiones previas que nos ayudarán en el diseño y desarrollo del proyecto.

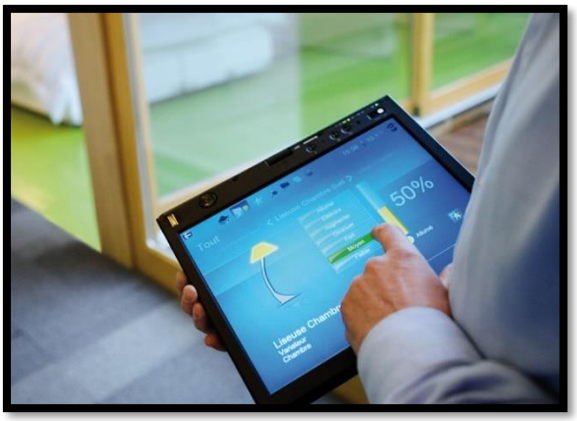


Figura 1. Domotización

RANCH SYSTEMS trata de una empresa de tecnología de alta calidad orientada a soluciones y aplicaciones con tecnología inalámbrica para el control y monitorización de espacios naturales y medio ambiente (2).

Figura 2. Ranch Systems

Trabajan con sensores de calidad y desarrollan soluciones de control y monitorización Cloud con aplicaciones de visualización de los datos muy completas que ofrecen una gran experiencia al usuario sin importar el dispositivo (2).

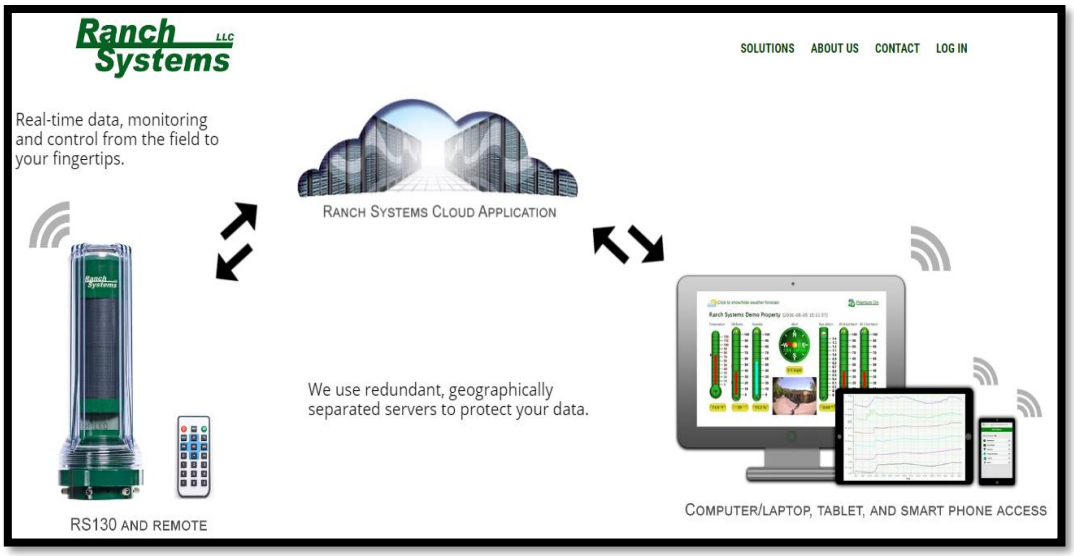


Figura 3. Ranch Systems. Comunicación

SAMCLA es una empresa centrada en la Telegestión del agua del riego de zonas verdes en espacios públicos y privados (3).

Figura 4. Samcla

Dentro de los servicios que ofrece destacamos "SmartHome System" basada en la telegestión residencial, nos ofrece gran cantidad de productos para poder domotizar nuestro jardín, nuestra vivienda u otros lugares y Software, para poder controlar su estado y acciones de manera sencilla, aplicaciones multiplataforma y con conectividad wifi (3).

Figura 5. Telegestión Residencial. Samcla

LIBELIUM es una empresa especializada en el sector de IoT, comercializa productos muy completos para multitud de casos de usos en distintos sectores. Ofrecen material de muy alta calidad a lo largo de todas las áreas implicadas en sus productos IoT, desde sensores y dispositivos de comunicación, hasta plataformas Cloud o aplicaciones personalizables donde gestionar tu sistema (4).

Figura 6. Libelium

Algunos de los productos que ofrece son muy interesantes, módulos de comunicación y conjuntos de sensores de alta calidad integrados en cajas aislantes con grandes acabados que permiten una instalación cómoda y sencilla. Además, cuenta con un catálogo de soluciones ya desarrolladas impresionantes con una gestión en la nube personalizable a gusto del cliente.

También disponen de material de calidad media-alta, disponible para compra de particulares para desarrollo de sistemas propios. Sin embargo, sigue siendo un coste algo elevado para los recursos económicos de los que se disponen en este proyecto.

A pesar de ser productos fuera del rango económico de nuestro proyecto, ha sido interesante y muy productivo investigar acerca de esta empresa y sus soluciones. Existe información muy valiosa y útil acerca de tecnología, diseños y arquitectura alrededor del negocio IoT (Internet of Things) de máxima calidad que puede ser de gran ayuda en un futuro (4).

2.3. Análisis de materiales

En este apartado se muestra el estudio del mercado actual en cuanto a los materiales que creemos necesarios para el correcto funcionamiento del producto.

2.3.1. Microcontrolador

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica.

Un microcontrolador incluye en su interior las tres principales unidades funcionales de un ordenador: CPU (Unidad Central de Procesamiento), memoria y periféricos de entrada salida.

Arduino Uno

Arduino Uno es una placa basada en el microcontrolador modelo ATMEGA328 (4). Posee 14 pines de entradas/salidas digitales (6 de los cuáles su pueden emplear como salidas PWM (Modulación del Ancho de Pulso) y dos de ellas se pueden emplear para la comunicación serie) y 6 entradas analógicas, un oscilador de 16MHz, conexión USB (Universal Serial Bus), conector de alimentación, un puerto ICSP (Programación Serial en Circuito), y un botón de reinicio (4).

Es posible consultar las características completas de esta placa y otras en la página oficial del fabricante (4).

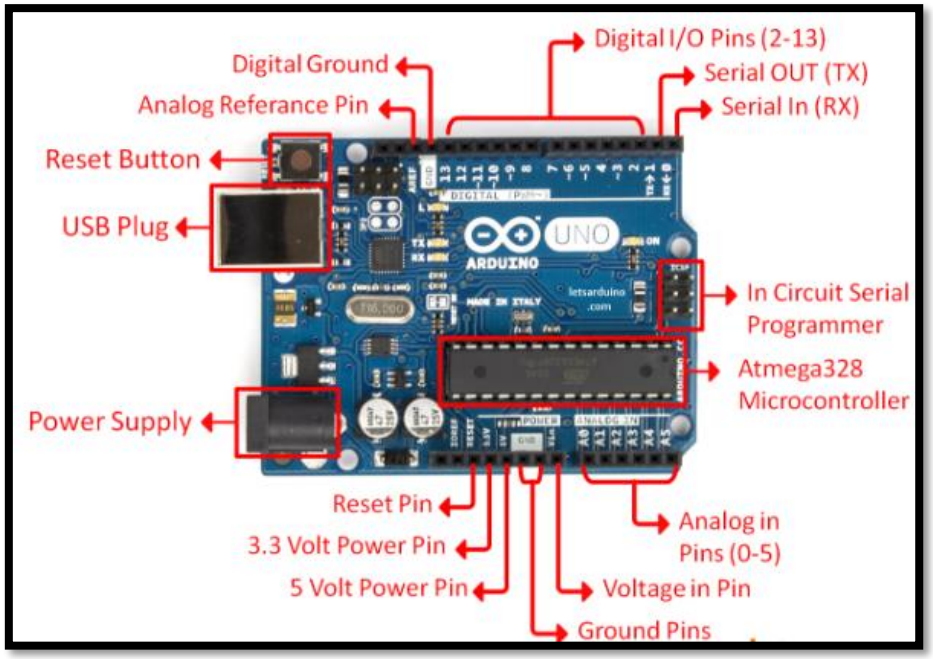


Figura 7. Placa Arduino Uno

Arduino Mega 2560

Arduino Mega 2560 es microcontrolador basado en el modelo ATmega2560. Cuenta con 54 pines digitales de entrada / salida (de los cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de Hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un puerto ICSP, Y un botón de reinicio (4).

Arduino YUN

Se trata de una placa Arduino integrada con un módulo wifi, salida de ethernet, entrada micro USB y adaptador microSD. Además, cuenta con un host USB para conectar memorias Flash, cámaras y otros dispositivos. Al tener numerosas especificaciones, su coste es elevado (4).

Figura 8. Arduino YUN

NodeMCU ESP8266 1.0 Amica

NodeMCU son una iniciativa open Source para el desarrollo de un modelo sencillo de integrar IoT (Internet of Things). Para ello desarrollan modelos de Hardware y Software que faciliten el desarrollo de programas y aplicaciones. Este modelo incluye un conector micro-USB para programar el chip interno y modulo wifi integrado para la comunicación inalámbrica (5).

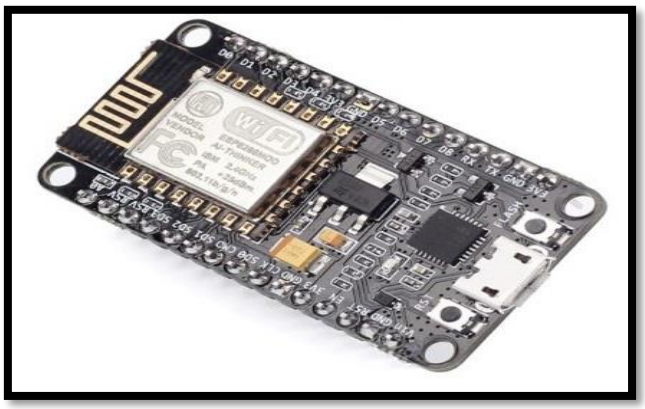


Figura 9. Prometec

2.3.2. Sensores

Un sensor es un objeto capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Analizamos los sensores existentes, que de alguna manera podrían aportar valor a nuestro sistema por estar relacionados con el medio ambiente o la tierra.

DHT11 y DHT22

El DHT11 es un sensor que proporciona una salida de datos digital. Entre sus ventajas podemos mencionar el bajo coste y el despliegue de datos digitales. Esto supone una gran ventaja frente a los sensores del tipo analógicos, en los cuales las fluctuaciones en el voltaje alteran la lectura de datos (6).

Entre las desventajas, el DHT11 solo lee enteros, no podemos leer temperaturas con decimales por lo que tenemos que pensarlo muy bien a la hora de utilizar este sensor para trabajos en los que se requieran lecturas precisas de temperatura y/o humedad (6).

El DHT22 es más preciso y posee mayor rango de medidas.

Tabla 1. Rangos de Medición y Precisión de Humedad y Temperatura

MODELO	**DHT11**	**DHT22**
Rango de medición de humedad	20-90 % HR	0-100% HR
Rango de medición de temperatura	0 hasta 50º C	-40 hasta 80º C
Precisión de humedad	± 5% HR	± 2 % HR
Precisión de temperatura	± 2º C	± 0. 5º C

STH25

El SHT25 es un circuito sensor de humedad y temperatura de alta gama en un encapsulado DFN (trazadores de pistas) de 6 pines. Este dispositivo proporciona señales calibradas en formato I2C (6). El sensor SHT25 contiene un sensor de humedad de tipo capacitivo, un sensor de temperatura de banda prohibida y un circuito integrado especializado de analógico a digital en

un único chip CMOSens. Rendimiento de sensor elevado en términos de precisión y estabilidad, así como un consumo de energía mínimo (6).

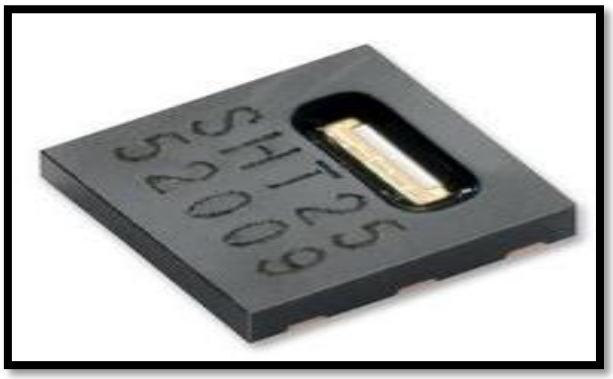


Figura 10. STH25. Sensor humedad y temperatura

GL5537

Fotorresistencia LDR (Light Dependent Resistor) o resistencia dependiente de la luz o fotocélula. Es una resistencia que varía su resistencia en función de la luz que incide sobre su superficie. Cuanto mayor sea la intensidad de la luz que incide en la superficie del LDR menor será su resistencia y cuanta menos luz incida mayor será su resistencia. (GL5528, GL5537, GL5539).

Figura 11. Resistencia Luminosa

FC-28 y HL-69

Un sensor sencillo que mide la humedad del suelo por la variación de su conductividad. No tiene la precisión suficiente para realizar una medición absoluta de la humedad del suelo, pero tampoco es necesario para controlar un sistema de riego (7).

Los valores obtenidos van desde 0 sumergido en agua, a 1023 en el aire (o en un suelo muy seco). Un suelo ligeramente húmedo daría valores típicos de 600-700 (7). Un suelo seco tendrá

valores de 800-1023. El valor concreto dependerá del tipo de suelo y la presencia de elementos químicos, como fertilizantes.

Además, no todas las plantas requieren la misma humedad, por lo que lo mejor es hacer una pequeña calibración en el terreno real (7).

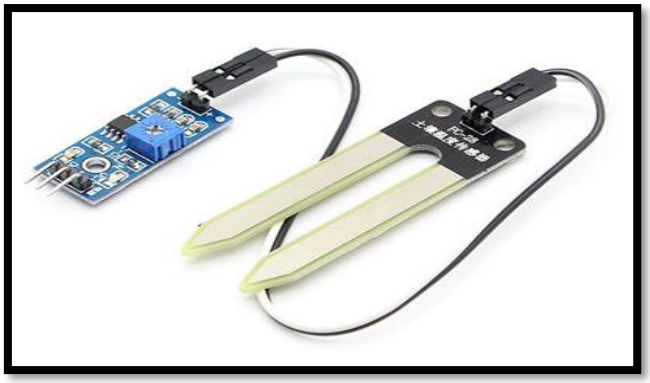


Figura 12. Sensor de humedad del suelo

YL-83

Este módulo consiste en una serie de pistas conductoras impresas sobre una placa de baquelita. La separación entre las pistas es muy pequeña (8). Lo que este módulo hace es crear un corto circuito cada vez que las pistas se mojan (8). El agua hace que se cree un camino de baja resistencia entre las pistas con polaridad positiva y las pistas conectadas al GND (Toma de Tierra). La corriente que fluye a través de estas pistas se ve limitada por resistencias de 10K en cada conductor, lo que impide que el corto circuito que se genera cuando se moja la placa vaya a estropear el micro controlador (8).

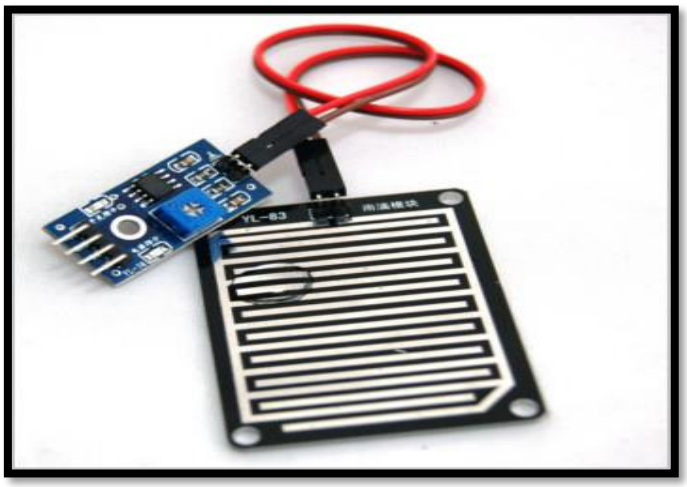


Figura 13. Detector de lluvia

YL-38

Hay sensores, como el YL-69 e YL-83, que simplemente transmiten una diferencia de potencial en sus medidas. El módulo YL-38 es el encargado de alimentar los otros módulos e interpretar esa diferencia de potencial medida por ellos. Para ello cuenta con un amplificador operacional (CI comparador LM393 SMD), que se encarga de amplificar la señal recibida y compararla. Aquí se genera la señal de salida que puede ser analógica (0-1023) y/o digital (0-1).

Este módulo cuenta además con resistencias limitadoras de corriente para no quemar el circuito y dos leds, uno de encendido-apagado y otro que indica la transmisión de datos por el pin de salida digital.

La señal digital funciona con un umbral de disparo a partir del cual cambiará su valor de 0 a 1 si lo superamos o, al contrario, de 1 a 0 si caemos de este umbral. Este umbral de sensibilidad se puede regular manualmente mediante el potenciómetro del circuito de control del YL-38.

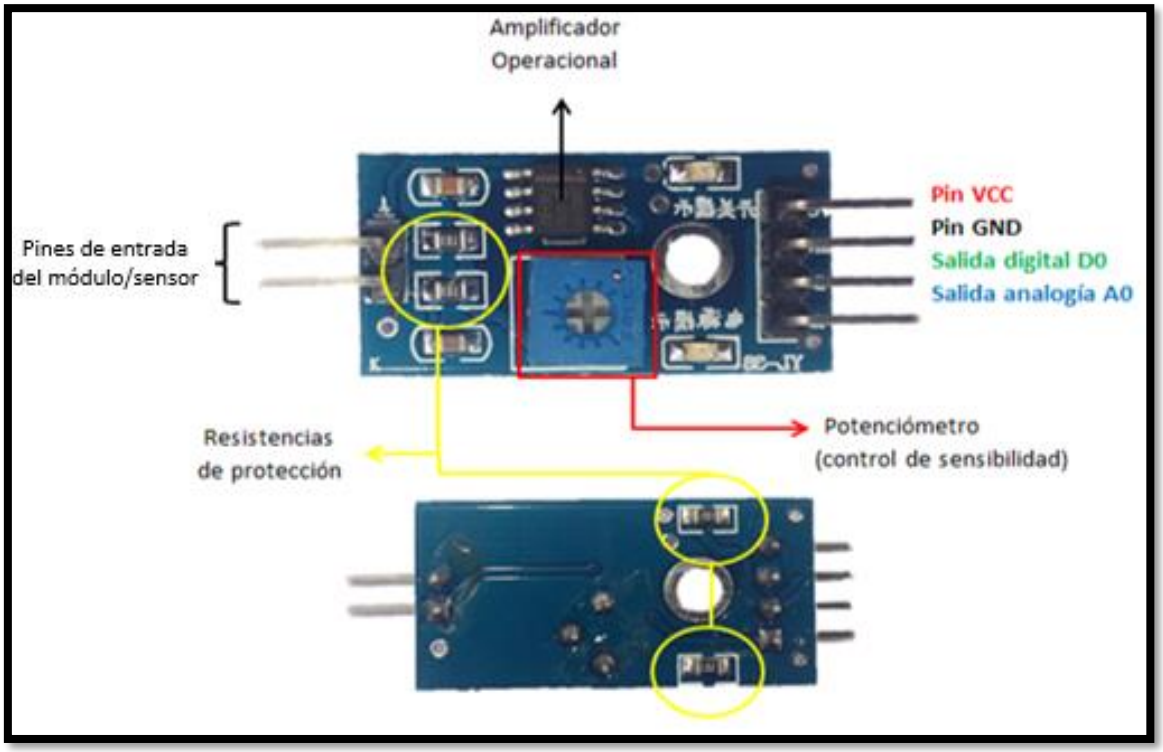


Figura 14. Módulo YL-38.

2.3.3. Comunicaciones

Bluetooth

Bluetooth es una especificación industrial para WPAN (Redes Inalámbricas de Área Personal) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM (Industrial, Ciencia y Medicina) de los 2.4 GHz.

Los principales objetivos que se pretenden conseguir con esta norma son (9):

- Facilitar las comunicaciones entre dispositivos móviles.
- Eliminar los cables y conectores.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

La especificación de Bluetooth define un canal de comunicación a un máximo 720 kbit/s (1 Mbit/s de capacidad bruta) con rango óptimo de 10 m (opcionalmente 100 m con repetidores) (9).

Opera en la frecuencia de radio de 2,4 a 2,48 GHz con amplio espectro.

Los saltos de frecuencia se dan entre un total de 79 frecuencias con intervalos de 1 MHz; esto permite dar seguridad y robustez (9).

ZigBee

ZigBee surge de la necesidad de desarrollar una tecnología inalámbrica de no muy alta transferencia de datos. Así, en 1998, un conjunto de empresas, conocidas como la ZigBee Alliance se juntaron para crear un estándar de comunicaciones que complementara a Wifi y Bluetooth (10).

El principal objetivo es conectar aplicaciones que requieren una comunicación segura, con baja tasa de envío y maximización de la vida útil de sus baterías. Se basa en dispositivos inalámbricos operando en la banda ISM para usos industriales, científicos y médicos (868 MHz, 915 MHz y 2.4 GHz) (10).

Tipos de dispositivos (según ZigBee) (10):

- **Coordinador**: Controla el ruteado y administra la red. Existe uno por red.
- **Router**: Interconecta diferentes nodos mediante direccionamiento.
- **Dispositivo final**: Elemento pasivo que responde ante peticiones de otros dispositivos. Se pasa la mayor parte del tiempo dormido.

Wifi

Es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica, pudiendo conectarse a internet a través de un punto de acceso de red inalámbrica.

Esta nueva tecnología surgió por la necesidad de establecer un mecanismo de conexión inalámbrica que fuese compatible entre distintos dispositivos.

Existen varias alternativas para garantizar la seguridad de estas redes. Las más comunes son la utilización de protocolos de cifrado de datos para los estándares wifi como el WEP, el WPA, o el WPA2 que se encargan de codificar la información transmitida para proteger su confidencialidad, proporcionados por los propios dispositivos inalámbricos.

2.3.4. Alimentación

A la hora de diseñar una red inalámbrica de sensores, se invierte mucho tiempo en resolver desafíos de conectividad y en pensar en las diferentes opciones de tecnología inalámbrica existentes para emplear la mejor de todas. Sin embargo, la alimentación es un elemento del sistema tan importante como el resto. La elección de una batería adecuada puede determinar el éxito o fracaso de un proyecto (11).

La capacidad total de una batería se mide en mili-amperios/hora (mAh) o amperios/hora (Ah). Una capacidad de 3Ah significa que la batería puede suministrar 3A durante una hora o 1A durante 3 horas. También es importante especificar cuándo una batería está vacía, esto es, cuando la tensión cae por debajo de cierto nivel necesario en el exterior (11).

Normalmente existen dos criterios importantes a la hora de elegir una batería. Tal y como se expone en [W5] (11):

- El primero tiene que ver con la temperatura. Las baterías frías conservan bien la electricidad y las baterías calientes suministran bien la electricidad, por tanto, es bueno conservar la batería en un lugar a baja temperatura y calentarla cuando se vaya a suministrar energía. Este hecho cobra importancia en una aplicación como la nuestra pues tratamos con sensores de temperatura.

  En principio no expondremos los sensores a temperaturas extremas y normalmente se tomarán datos a temperatura ambiente, por lo que se pasará por alto el diferente comportamiento de las baterías según la temperatura, sin embargo, es importante no olvidarlo para futuros trabajos (11).
- La segunda regla se refiere a la no linealidad de las baterías. La capacidad de una batería está relacionada con la corriente que se le pide suministrar y esta no es lineal. Un alto consumo de corriente hace disminuir la capacidad de la batería. Suponer que una batería es lineal para toda corriente es peligroso, y solo es cierto cuando los cambios de temperatura y de corriente son muy pequeños (11).

Además, habrá que tener en cuenta aspectos como los voltajes de alimentación de los módulos Xbee (módulos de comunicación por radio) (2.8 – 3.4 V) y el tamaño de la batería (11).

Baterías alcalinas.

Las baterías alcalinas son las más comunes en hogares y son idóneas para una gran variedad de aplicaciones electrónicas, sobre todo porque las empresas de baterías han mejorado la química en los últimos años (11).

Las pilas alcalinas son de bajo costo, ampliamente disponibles y son ideales para aplicaciones de baja corriente a temperatura ambiente, pero tienen dos defectos: son muy malas en condiciones frías y no tienden a funcionar correctamente en condiciones de alta corriente (11).

Baterías de litio.

Las variedades de baterías de litio son muchas. Es complicado clasificar a todas en un mismo grupo que siga un modelo común, pero sí tienen unas características comunes. Son baterías desechables y son mejores que las alcalinas a bajas temperaturas (11).

Baterías de níquel-metal hidruro (NiMH).

Es una de las mejores baterías recargables. La batería NiCad (níquel-cadmio), tiende a mantener sus características tras muchas recargas. Es la batería alcalina de las recargables y su comportamiento en capacidad sigue un modelo similar descrito por un polinomio de segundo orden, con un mejor proceder a bajas temperaturas y no disminuyendo tan rápidamente la capacidad a medida que aumenta la demanda de corriente (11).

Tabla 2. Ejemplos baterías

ALCALINAS	LITIO	NiMH
		

2.3.5. Electroválvula

Una electroválvula es una válvula electromecánica, diseñada para controlar el paso de un fluido por un conducto o tubería. La válvula se mueve mediante una bobina solenoide. Generalmente no tiene más que dos posiciones: abierto y cerrado (12).

2.3.6. Modulo wifi

ESP8266

Se trata de un chip integrado con conexión Wifi y compatible con el protocolo TCP/IP (13).

Hardware (13).

- Utiliza una CPU Tensilica L106 32-bit
- Voltaje de operación entre 3V y 3,6V
- Corriente de operación 80 mA
- Temperatura de operación -40ºC y 125ºC

Conectividad (13).

- Soporta IPv4 y los protocolos TCP/UDP/HTTP/FTP.
- No soporta HTTPS en un principio. Sí que lo hace mediante Software tanto en cliente como servidor TLS1.2. La primera implementación está todavía en desarrollo.

Wifi shield

Se trata de una placa capaz de adaptarse a la mayoría de microcontroladores, que aporta la capacidad de conectividad inalámbrica a través de wifi o salida ethernet dependiendo el modelo. También las hay que integran otras capacidades extras como adaptadores microSD (14).

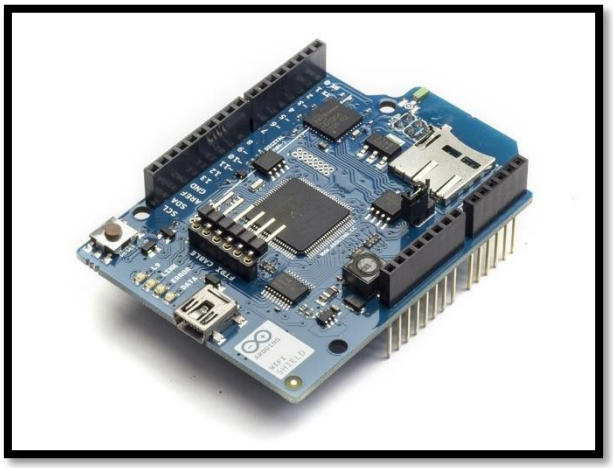


Figura 15. Placa Wifi Shield

2.4. Análisis de las tecnologías para el desarrollo del proyecto.

En este apartado veremos los distintos entornos de desarrollo que se encuentran en la actualidad para realizar el desarrollo de la parte Software del proyecto.

Android Studio

Android Studio es un entorno de desarrollo integrado para la plataforma Android. Está disponible para desarrolladores. Este entorno está basado en IntelliJ IDEA de JetBrains, diseñado específicamente para desarrollar aplicaciones para Android. Está disponible para descargar para Windows, Mac OS X y Linux (15).

Incluye herramientas completas de edición, depuración, pruebas y perfilamiento de códigos, y funcionalidades para mejorar la programación, usabilidad, rendimiento y otros problemas orientado en Android (15).

Android Studio ofrece:

- Sistema de construcción
- Ayudas para la codificación
- Pre visualización de recursos
- Generación de recursos
- Detección de errores
- Debugging
- Refactorización
- Ayudas paa el diseño
- Acceso a servicios Google

Figura 16. Android Studio

Xcode

Xcode es el entorno de desarrollo integrado de Apple Inc. Está disponible de forma gratuita en la Mac App Store, y es una opción ideal para los desarrolladores que están trabajando en el siempre creciente mercado de aplicaciones para Mac, iPhone y iPad. Nos permite usar una multiplicidad de herramientas que nos permiten incrementar la productividad, con una interfaz fácil y la posibilidad de hacer Debugging (16).

Xcode ofrece:

- Espacios de trabajo o Works paces, con la finalidad de poder subdividir las prtes de una aplicacion
- Ayudas para la codificación
- Debugging
- Detección de errores
- Refactorización

Figura 17. Xcode

Visual Studio

Visual Studio 2015 es un completo entorno de desarrollo integrado para crear aplicaciones espectaculares para Windows, Android e iOS, además de aplicaciones web y servicios de nube innovadores (17).

Visual Studio ofrece:

Herramientas y servicios para proyectos de cualquier tamaño o complejidad
Programar en: C#, Visual Basic, F#, C++, Python, Node.js y HTML/JavaScript
Planificación de sprint
Depuración y creación de perfiles avanzadas, pruebas automatizadas y manuales
DevOps con implementaciones automatizadas y supervisión continua.
Ayudas para la codificación.
Detección de errores.
Depuración.
Refactorización

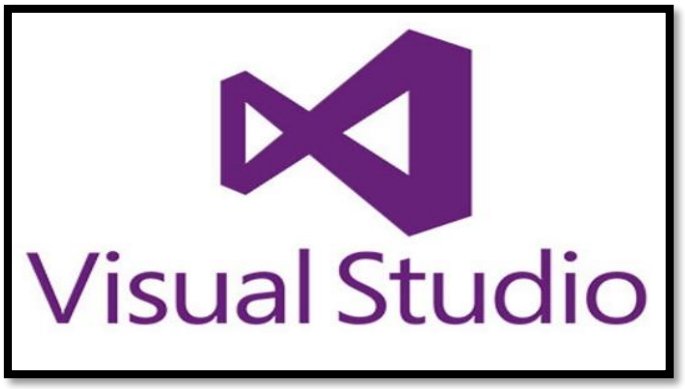


Figura 18. Visual Studio

Eclipse

Eclipse es un entorno de desarrollo multiplataforma de código abierto, para el desarrollo de aplicaciones de escritorio web y móviles (18).

Eclipse ofrece:

- Excelente gestión de Proyectos.
- Soporte para plugins.
- Opciones de personalización.
- Permite la integración de soluciones de terceros.
- Ofrece herramientas para repositorios y servidores.
- Soporta lenguajes como Java, C, C++, JSP, perl y php gracias a la amplia variedad de plugins disponibles.
- Disponible para Windows, Linux y Mac.
- Ayudas para la codificación.
- Detección de errores.
- Perspectiva específica y vista de errores.
- Depuración.
- DESVENTAJAS --> Mayor consumo de recursos

Figura 19. Eclipse

Arduino IDE

Arduino dispone de un entorno de desarrollo propio desarrollado en Java y está pensada para fomentar y facilitar el uso de la electrónica para el público en general, puede ser utilizado con cualquier placa Arduino y otras placas externas a la marca y sistemas operativos de Windows, Mac OS y Linux (19).

Figura 20. Arduino

Thinger.io

Es una plataforma libre para trabajar con internet de las cosas (IoT). La consola web (Cloud Console) permite controlar, configurar y administrar tus dispositivos y visualizar la información en la nube. La plataforma Thinger esta soportada por una REST API para integrar de forma sencilla a otras aplicaciones, como servicios web, móviles o aplicaciones de escritorio (20).

Thinger ofrece:
Codigo libre.
Multidispositivos.
Plataforma en la nube.
Codigo facil de integrar.
Para individuos.
Para compañias.

Figura 21. Thinger.io

XCTU

XCTU es una aplicación multi-plataforma gratuita diseñada para permitir a los desarrolladores interactuar con los módulos RF de Digi a través de una interfaz gráfica sencilla de usar. Incluye herramientas que facilitan la configuración y prueba de los módulos XBee® RF.

XCTU incluye todas las herramientas que necesita un desarrollador para ponerse rápidamente en funcionamiento con XBee. Las características únicas, como la representación gráfica de la red, que representan gráficamente la red XBee junto con la intensidad de la señal de cada conexión, y el constructor de tramas de la API de XBee, que intuitivamente ayuda a construir e interpretar los marcos de API para XBees que se usan en modo API, En la plataforma XBee más fácil que nunca (21).

XCTU ofrece:

- Puede administrar y configurar varios dispositivos de RF, incluso dispositivos conectados remotamente (por aire).
- El proceso de actualización del firmware restaura sin problemas los ajustes del módulo, el modo de manejo automático y los cambios en la velocidad en baudios.
- Dos consolas API y AT específicas, han sido diseñadas desde cero para comunicarse con sus dispositivos de radio.
- XCTU incluye un conjunto de herramientas integradas que se pueden ejecutar sin tener ningún módulo RF conectado:
- Generador de cuadros: genera fácilmente cualquier tipo de marco API para guardar su valor.
- Interpretador de marcos: Descodifica una trama de API y ve sus valores de trama específicos.
- Recuperación: Recuperar los módulos de radio que han dañado el firmware o están en modo de programación.
- Cargar sesión de consola: Carga una sesión guardada en cualquier PC que ejecute XCTU.
- Prueba de rango: Realiza una prueba de rango entre 2 módulos de radio de la misma red.
- Explorador de firmware: Navega a través de la biblioteca de firmware de XCTU.
- Un proceso de actualización le permite actualizar automáticamente la propia aplicación y la biblioteca de firmware de radio sin necesidad de descargar ningún archivo adicional.
- XCTU contiene documentación completa a la que se puede acceder en cualquier momento.

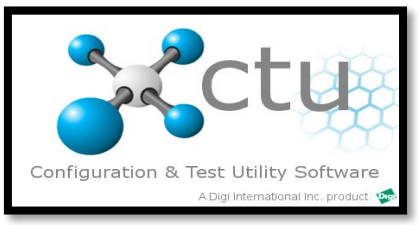


Figura 22. CTU

2.5. Elección de las tecnologías

La idea es contar con una aplicación de escritorio con la que el usuario pueda acceder a todos los servicios ofrecidos e interactuar con ellos, aunque se valorará la posibilidad de crear más adelante una aplicación móvil para Android.

Una buena elección sería Eclipse, que nos ofrece amplias posibilidades en nuestro desarrollo y herramientas como WindowBuilder (22) con la que somos capaces de desarrollar una interfaz de usuario GUI (Interfaz de Usuario Gráfica) de manera sencilla y rápida gracias a las facilidades que ofrece en el desarrollo de este tipo de aplicaciones, añadiendo controles con sus funciones de drag-and-drop, añadiendo eventos generando el código automáticamente. Además, ya se han realizado previamente proyectos con estas herramientas, por lo que el coste de aprendizaje es prácticamente nulo.

Otra opción sería un servidor web en el que instalar una aplicación con visualización html5, con CSS3 y código bootstrap con el que adaptar la interfaz a los distintos dispositivos.

Sin embargo, después de haber estudiado las posibilidades que ofrece Thinger, sus funcionalidades y servicios, sería ilógico obviar esta herramienta existente que nos brinda tantas facilidades. Por ello, vamos a omitir en primera instancia el apartado de desarrollo de aplicación o interfaz gráfica, utilizando para ello la plataforma web **Thinger** (20).

Para el desarrollo del controlador NodeMCU, utilizaremos el entorno de desarrollo facilitado por la marca Arduino (**Arduino Software IDE**) (19).

3. Desarrollo del proyecto

3.1. Descripción general

3.1.1. Perspectiva del producto

Tras realizar el estudio de mercado de las diferentes empresas y servicios que ofrecen y analizar los componentes más relevantes existentes para la elaboración de nuestro sistema, podemos empezar a desarrollar nuestro proyecto con las conclusiones obtenidas de una manera óptima y acorde con nuestras necesidades y limitaciones, tanto tecnológicas como económicas.

El sistema de riego consiste en controlar los parámetros medioambientales de temperatura, humedad y luminosidad y, en función de los niveles, ser capaz de activar o desactivar las electroválvulas para proceder al riego de la zona.

El sistema tendrá tres formas de funcionamiento:

- **Funcionamiento manual**: Permite al usuario controlar de forma remota la activación o desactivación de las electroválvulas.
- **Funcionamiento automático**: El sistema actúa en consecuencia a los niveles humedad del suelo recibidos, si estos se encuentran por debajo de un umbral definido activará las electroválvulas automáticamente, si las condiciones climáticas son adecuadas (baja temperatura, sin precipitación, luminosidad baja, y horario adecuados).
- **Funcionamiento programado**: El sistema actúa en consecuencia al horario establecido por el usuario y activará las electroválvulas automáticamente el tiempo de riego establecido.

Mediante la placa NodeMCU (23), controladores Zigbee (10), sensores, electroválvulas y otros componentes se simulará el funcionamiento de un sistema de riego real. Se configurará una interfaz sirviéndonos de Thinger a través del cual se controlará su funcionamiento.

3.1.2. Alcance del Software

El sistema se centrará en dos elementos principales:

Sistema de monitorización: Software de monitorización, configuración y control a través de Thinger (o desarrollar una interfaz) con las funcionalidades necesarias para controlar el sistema de riego.

El objetivo principal está centrado en que el sistema facilite la gestión del riego, controlando los parámetros medioambientales y de esta manera otorgar más información acerca del estado del lugar y poder decidir con mayor conocimiento acerca de las acciones a llevar a cabo.

Sistema automático: Software de control automático que se encontrará programado en la placa NodeMCU (23). Este Software se desarrollará en el entorno de programación de Arduino, dicho sistema controlará de forma automática el encendido o apagado de las electroválvulas en función de los parámetros de establecidos y medidos por la placa y otros sensores transferidos por los módulos XBee (10).

3.1.3. Capacidades generales

Las capacidades principales que tendrá el sistema son:

- El sistema permite al usuario activar o desactivar el sistema automático y definir sus umbrales críticos a partir de los cuales el sistema deberá actuar de forma automática.
- La interfaz contará con un panel en el que se indicará al usuario el estado de la electroválvula y los parámetros medioambientales medidos.
- El sistema automático tiene por norma ceder ante cualquier acción manual ejecutada por el usuario, abortando su actuación automática.
- En el momento en el que falle la comunicación entre el sistema y actuadores o sensores, el sistema automático quedará inutilizado sin producir ningún comportamiento inseguro.

3.1.4. Restricciones generales

El sistema deberá ser tolerante a fallos que deberán estar controlados, ya que cualquier fallo del sistema supondría un grave peligro de inundación o sequía que, en cualquier caso, comprometería el estado del ecosistema del lugar.

El uso correcto del sistema de riego de forma manual queda delegado totalmente en el usuario, quien tendrá completa libertad y será responsable de su correcto uso.

El sistema automático deberá configurarse con valores de umbrales inferiores o iguales a 25% de humedad de tierra (15% por defecto).

No se permitirá la activación de las electroválvulas en modo automático si las condiciones no son óptimas para el riego. Se consideran condiciones desfavorables las temperaturas y luminosidad elevadas (temp: >= 20ºC, lum: >= 500), un horario de 11:00 a 18:00 o precipitación.

No se permitirá la activación de las electroválvulas en modo programado si la humedad de la tierra es superior al 60% o existe precipitación en ese momento.

El sistema nunca podrá actuar en modo automático o programado sin que un usuario sea quien apruebe o configure dichas acciones (por defecto desactivadas).

3.1.5. Entorno operacional

El entorno de ejecución de la aplicación de configuración y control del sistema será un ordenador con un Sistema Operativo Windows 7 o superior y un entorno de java jdk-8u60 o compatibles en Eclipse y conexión a internet.

Por otro lado, el código que transformará las distintas señales de los actuadores en acciones se llevará a cabo en un dispositivo físico NodeMCU ESP8266 1.0 Amica (23) y módulos ZigBee (10).

3.2. Análisis

En esta sección se realiza un análisis exhaustivo del sistema, definiendo los distintos casos de uso y requisitos que debe cumplir con la finalidad de alcanzar el objetivo deseado.

3.2.1. Casos de uso

En este apartado se especifican los casos de uso que representan las distintas funcionalidades del producto.

- Activar / Desactivar Riego
- Configurara tiempo de riego
- Activar / Desactivar modo automático
- Configurar umbral
- Activar / Desactivar programación de riego
- Configurar hora de riego
- Comprobar indicadores climáticos
- Ejecución modo automático
- Ejecución programación
- Envío de alertas

Los casos de uso serán representados en una tabla con la siguiente estructura.

Tabla 3. Ejemplo modelo tabla de casos de uso

Identificador CU_XX	
Nombre del caso de uso	
Actor	
Propósito	
Flujo	

- Identificador: el requisito es identificado por CU_XX, donde XX corresponde al caso de uso.
- Nombre: nombre descriptivo del caso de uso.
- Actor: actores que operan en el caso de uso.
- Propósito: descripción breve del caso de uso.
- Flujo: pasos de ejecución del caso de uso.

Tabla 4. Identificador CU_01

Identificador CU_01	
Nombre del caso de uso	Activar / Desactivar riego
Actor	Usuario
Propósito	Poder activar o desactivar manualmente la electroválvula de riego a través de la interfaz.
Flujo	1. En la plataforma Thinger, seleccionamos “Dashboards”. 2. Seleccionamos el apartado de Control. 3. Tendremos un botón de acción llamado “Electroválvula” para activar o desactivarla.

Tabla 5. Identificador CU_02

Identificador CU_02	
Nombre del caso de uso	Configurar tiempo de riego
Actor	Usuario
Propósito	Poder establecer el tiempo en minutos (máx. 30) que transcurre desde que se activa la electroválvula para proceder al riego hasta que se cierra automáticamente si no se ha cerrado previamente. Es un temporizador que actúa como medida de seguridad, y por defecto está a 10 min.
Flujo	1. En la plataforma Thinger, seleccionamos "Dashboards". 2. Seleccionamos el apartado de Configuración. 3. Tendremos una barra de valores (slider), llamada "Water time" para ajustar el tiempo de riego.

Tabla 6. Identificador CU_03

Identificador CU_03	
Nombre del caso de uso	Activar / Desactivar modo automático
Actor	Usuario
Propósito	Poder activar o desactivar manualmente el modo automático donde el sistema regará de manera autónoma si fuera necesario durante el tiempo de riego establecido, a partir de los valores recogidos por los sensores y los umbrales fijados.
Flujo	1. En la plataforma Thinger, seleccionamos "Dashboards". 2. Seleccionamos el apartado de Control. 3. Tendremos un botón de acción llamado "Modo automático" para activar o desactivarlo.

Tabla 7. Identificador CU_04

Identificador CU_04	
Nombre del caso de uso	Configurar umbral
Actor	Usuario
Propósito	Poder establecer los umbrales críticos de los sensores, a partir de los cuales si son superados todos ellos actuarán el sistema de manera autónoma si el modo automático está activo y las condiciones climáticas son favorables.
Flujo	1. En la plataforma Thinger, seleccionamos "Dashboards". 2. Seleccionamos el apartado de Configuración. 3. Tendremos una barra de valores (slider), llamada "Umbral de tierra" para fijar el umbral (XX%).

Tabla 8. Identificador CU_05

Identificador CU_05	
Nombre del caso de uso	Activar / Desactivar programación de riego
Actor	Usuario
Propósito	Poder activar o desactivar manualmente la programación de riego mediante el cual, el sistema regará de forma automática en la hora y minuto configurados durante el tiempo de riego.
Flujo	1. En la plataforma Thinger, seleccionamos "Dashboards". 2. Seleccionamos el apartado de Control. 3. Tendremos un botón de acción llamado "Programación" para activar o desactivarlo.

Tabla 9. Identificador CU_06

Identificador CU_06	
Nombre del caso de uso	Configurar hora de riego
Actor	Usuario
Propósito	Poder establecer una determinada hora y minuto en la que regar si el modo de programación está activado.
Flujo	1. En la plataforma Thinger, seleccionamos “Dashboards”. 2. Seleccionamos el apartado de Configuración. 3. Tendremos dos barras de valores (sliders), llamadas “Hour” para fijar la hora y “Minute” para establecer el minuto.

Tabla 10. Identificador CU_07

Identificador CU_07	
Nombre del caso de uso	Comprobar indicadores climáticos
Actor	Usuario
Propósito	Poder acceder a la información recogida por los sensores y actuadores del sistema de manera visual a través de la interfaz a modo de graficas o de manera más detallada numéricamente.
Flujo	1. En la plataforma Thinger, seleccionamos “Dashboards”. 2. Seleccionamos el apartado de Monitorización. 3. Tendremos las gráficas de los valores de los parámetros climáticos.
	1. En la plataforma Thinger, seleccionamos “Data Buckets”. 2. Tendremos las tablas de información de los parámetros climáticos.

Tabla 11. Identificador CU_08

Identificador CU_08	
Nombre del caso de uso	Ejecución modo automático
Actor	Sistema
Propósito	Si el sistema se encuentra en modo automático, debe poder activar y desactivar la electroválvula de manera autónoma si se cumplen los valores de umbrales fijados por el usuario, y las condiciones atmosféricas.
Flujo	No aplica.

Tabla 12. Identificador CU_09

Identificador CU_09	
Nombre del caso de uso	Ejecución programación
Actor	Sistema
Propósito	Si el sistema se encuentra en modo programación, debe poder activar y desactivar la electroválvula de manera autónoma si se cumple el horario establecido.
Flujo	No aplica.

Tabla 13. Identificador CU_10

Identificador CU_10	
Nombre del caso de uso	Envío de alertas
Actor	Sistema
Propósito	Si el sistema se activa en modo programación o modo automático, debe enviar un email de aviso al usuario de manera autónoma con información al respecto.
Flujo	No aplica.

3.2.2. Requisitos

En primer lugar, hemos definido los requisitos funcionales que definen las funciones del sistema.

En segundo lugar, hemos definido los no funcionales ya que complementan los casos de uso (Configuración, interfaz y lenguajes de programación).

Los requisitos tanto funcionales como no funcionales, serán representados en una tabla con la siguiente estructura.

Tabla 14. Ejemplo modelo tabla de requisitos

Identificador	
Nombre del requisito	Nombre …
Prioridad	Alta || Media || Baja
Descripción	Descripción …

- Identificador: el requisito será identificado por RF_XX (Requisitos funcionales) y RNF_XX (Requisitos no funcionales) donde XX corresponde al número de requisito.
- Nombre: nombre descriptivo del requisito.
- Prioridad: grado de prioridad del requisito.
- Descripción: breve descripción del objetivo del requisito.
- Prueba: se incluirán las pruebas asociadas al requisito.

Requisitos funcionales.

Tabla 15. Requisitos funcionales. Identificador RF_01

Identificador RF_01	
Nombre del requisito	Activación manual de las electroválvulas
Prioridad	Alta
Descripción	El usuario podrá activar siempre de manera manual las electroválvulas, si las condiciones no son óptimas se le comunicarán con un aviso. (Elevada temperatura y luminosidad, o lloviendo actualmente).

Tabla 16. Requisitos funcionales. Identificador RF_02

Identificador RF_02	
Nombre del requisito	Desactivación manual de las electroválvulas
Prioridad	Alta
Descripción	El usuario podrá desactivar las electroválvulas de forma manual, bajo cualquier condición.

Tabla 17. Requisitos funcionales. Identificador RF_03

Identificador RF_03	
Nombre del requisito	Activación modo automático
Prioridad	Alta
Descripción	El usuario podrá activar el modo automático, de esta manera el sistema funcionará de forma autónoma.

Tabla 18. Requisitos funcionales. Identificador RF_04

Identificador RF_04	
Nombre del requisito	Activación de electroválvulas en modo automático
Prioridad	Alta
Descripción	El sistema será capaz de activar las electroválvulas de forma automática si la situación lo requiere durante el tiempo de riego definido en el sistema. Esta activación vendrá determinada por el valor actual de humedad de la tierra que recoge el sensor, su umbral establecido, las condiciones climáticas restrictivas y el horario.

Tabla 19. Requisitos funcionales. Identificador RF_05

Identificador RF_05	
Nombre del requisito	Tiempo de riego
Prioridad	Alta
Descripción	El sistema controlará el tiempo de riego (activación de electroválvulas) en función de los minutos establecidos de riego (máx. 30 minutos), mediante la integración de una biblioteca que controle el horario.

Tabla 20. Requisitos funcionales. Identificador RF_06

Identificador RF_06	
Nombre del requisito	Desactivación de electroválvulas en modo automático
Prioridad	Alta
Descripción	El sistema deberá desactivar las electroválvulas después de cumplirse el tiempo establecido de riego.

Tabla 21. Requisitos funcionales. Identificador RF_07

Identificador RF_07	
Nombre del requisito	Tiempo de desactivación manual
Prioridad	Alta
Descripción	Si el sistema supera el tiempo de riego establecido, aunque no se encuentre en modo automático y haya sido el usuario el que accionó la electroválvula, está se cerrará automáticamente como medida de seguridad.

Tabla 22. Requisitos funcionales. Identificador RF_08

Identificador RF_08	
Nombre del requisito	Activación de electroválvulas en modo programación
Prioridad	Alta
Descripción	El sistema será capaz de activar las electroválvulas de forma automática si se cumple el horario establecido por el usuario.

Tabla 23. Requisitos funcionales. Identificador RF_09

Identificador RF_09	
Nombre del requisito	Desactivación de electroválvulas en modo programación
Prioridad	Alta
Descripción	El sistema será capaz de desactivar las electroválvulas de forma automática tras cumplirse el tiempo de riego.

Tabla 24. Requisitos funcionales. Identificador RF_10

Identificador RF_10	
Nombre del requisito	Configuración de horario
Prioridad	Alta
Descripción	El usuario podrá configurar una hora y minuto determinada a la que actuará el modo programación.

Tabla 25. Requisitos funcionales. Identificador RF_08

Identificador RF_11	
Nombre del requisito	Configuración de umbral
Prioridad	Media
Descripción	El usuario podrá configurar el umbral de humedad de la tierra desde la interfaz de usuario (rango 0 – 25%).

Tabla 26. Requisitos funcionales. Identificador RF_010

Identificador RF_12	
Nombre del requisito	Historial
Prioridad	Media
Descripción	El usuario podrá consultar las últimas acciones llevadas a cabo de forma manual o automática.

Tabla 27. Requisitos funcionales. Identificador RF_011

Identificador RF_13	
Nombre del requisito	Consulta de datos (BD)
Prioridad	Media
Descripción	El usuario podrá consultar los parámetros recibidos de los sensores tanto de forma gráfica (Dashboards) como de forma cuantitativa y fechada (Data Buckets).

Tabla 28. Identificador RT_14

Identificador RF_14	
Nombre del requisito	Avisos
Prioridad	Baja
Descripción	El sistema deberá alertar al usuario en caso de que active el riego a causa del modo automático o por la programación establecida.

Requisitos no funcionales.

Tabla 29. Requisitos No funcionales. Identificador RF_01

Identificador RNF_01	
Nombre del requisito	Disponibilidad
Prioridad	Alta
Descripción	El sistema deberá tener una alta disponibilidad puesto que su función depende en gran medida de los fenómenos medioambientales que no pueden ser controlados por el usuario. Dependerá también de la conectividad del dispositivo desde el que se ejecute.

Tabla 30. Requisitos No funcionales. Identificador RF_02

Identificador RNF_02	
Nombre del requisito	Usabilidad
Prioridad	Alta
Descripción	La interfaz gráfica de usuario (GUI) será clara y sin sobrecarga de información para facilitar al usuario la interacción con el sistema. Se utilizarán símbolos y formas intuitivas que permitan comprender al usuario su función.

Tabla 31. Requisitos No funcionales. Identificador RF_03

Identificador RNF_03	
Nombre del requisito	Lenguaje placa microcontrolador
Prioridad	Alta
Descripción	El Software empotrado se desarrollará en el lenguaje de programación processing v2.0 (2013) en el entorno ofrecido por la comunidad Arduino.

Tabla 32. Requisitos No funcionales. Identificador RF_04

Identificador RNF_04	
Nombre del requisito	Tiempo de respuesta
Prioridad	Alta
Descripción	El sistema no debe tardar más de 10 segundos en responder a una solicitud.

Tabla 33. Requisitos No funcionales. Identificador RF_05

Identificador RNF_05	
Nombre del requisito	Comunicación wifi
Prioridad	Media
Descripción	La comunicación entre el sistema y el servidor Thinger debe ser permanente siempre que el suministro energético de las instalaciones donde se encuentra la placa sean buenas.

Tabla 34. Requisitos No funcionales. Identificador RF_06

Identificador RNF_06	
Nombre del requisito	Comunicación entre el Hardware (Zigbee)
Prioridad	Media
Descripción	El dispositivo Zigbee aislado puede estará en modo “sleep” y despertará cada media hora para transmitir valores de humedad de tierra.

Tabla 35. Requisitos No funcionales. Identificador RF_07

Identificador RNF_07	
Nombre del requisito	Calculo de valor de temperatura…
Prioridad	Baja
Descripción	Las operaciones realizadas con la biblioteca DHT22 darán las medidas de temperatura en ºC y humedad en %.

3.2.3. Pruebas

A continuación, se muestran las pruebas realizadas para confirmar el correcto funcionamiento del sistema y de las funcionalidades descritas en los requisitos.

Tabla 36. Prueba RF_01

Prueba1 RF_01	
Nombre	Activación manual de las electroválvulas
Precondición	Electroválvula desconectada, estado cerrada en Thinger, estado aux en código false.
Acción	Activar electroválvula a través del control de Thinger.
Post-condición	Comprobar electroválvula activa, estado abierto en Thinger y estado aux en código true.

Tabla 37. Prueba RF_02

Prueba1 RF_02	
Nombre	Desactivación manual de las electroválvulas
Precondición	Electroválvula conectada, estado abierto en Thinger y estado aux en código true.
Acción	Desactivar electroválvula a través del control de Thinger.
Post-condición	Comprobar electroválvula desactivada, estado cerrada en Thinger y estado aux en código false.

Tabla 38. Prueba RF_03

Prueba1 RF_03	
Nombre	Activación modo automático
Precondición	Modo automático desconectado, estado en código false.
Acción	Activar modo automático a través del control de Thinger.
Post-condición	Comprobar modo automático activo, estado modo en código true.

Tabla 39. Prueba RF_04

Prueba1 RF_04	
Nombre	Activación de electroválvulas en modo automático
Precondición	Modo automático conectado, estado modo en código true, electroválvula desconectada, estado cerrada en Thinger, estado aux en código false.
Acción	Forzar que se cumplan las condiciones de activación automática.
Post-condición	Electroválvula conectada, estado abierta en Thinger, estado aux en código true.

Tabla 40. Prueba RF_05

Prueba1 RF_05	
Nombre	Tiempo de riego
Precondición	Campo en Thinger muestra el valor actual.
Acción	Configurar nuevo tiempo de riego.
Post-condición	Variable timeR actualizada en código con nuevo valor. Campo en Thinger actualizado.

Tabla 41. Prueba RF_06

Prueba2 RF_05	
Nombre	Tiempo de riego erróneo
Precondición	Campo en Thinger muestra el valor actual.
Acción	Configurar nuevo tiempo de riego erróneo. (menor de 1 o mayor de 30) o nulo.
Post-condición	Variable timeR no actualizada en código. Campo en Thinger se mantiene.

Tabla 42. Prueba RF_07

Prueba1 RF_06	
Nombre	Desactivación de electroválvulas en modo automático
Precondición	Modo automático conectado, estado modo en código true, electroválvula conectada, estado abierta en Thinger, estado aux en código true.
Acción	Cumplir tiempo de riego
Post-condición	Electroválvula desconectada, estado cerrada en Thinger, estado aux en código false.

Tabla 43. Prueba RF_08

Prueba1 RF_07	
Nombre	Tiempo de desactivación manual
Precondición	Electroválvula conectada, estado abierta en Thinger, estado aux en código true.
Acción	Cumplir tiempo de riego sin cerrar electroválvula previamente.
Post-condición	Electroválvula desconectada, estado cerrada en Thinger, estado aux en código false.

Tabla 44. Prueba RF_09

Prueba1 RF_08	
Nombre	Activación de electroválvulas en modo programación
Precondición	Modo programación conectado, estado program en código true, electroválvula desconectada, estado cerrada en Thinger, estado aux en código false.
Acción	Forzar que se cumpla una hora y minuto determinado
Post-condición	Electroválvula conectada, estado abierta en Thinger, estado aux en código true.

Tabla 45. Prueba1 RF_10

Prueba1 RF_09	
Nombre	Desactivación de electroválvulas en modo programación
Precondición	Modo programación conectado, estado program en código true, electroválvula conectada, estado abierta en Thinger, estado aux en código true.
Acción	Cumplir tiempo de riego.
Post-condición	Electroválvula desconectada, estado cerrada en Thinger, estado aux en código false.

Tabla 46. Prueba1 RF_10

Prueba1 RF_10	
Nombre	Configuración de horario
Precondición	Campos en Thinger muestran los valores actuales.
Acción	Introducir nueva hora y minuto.
Post-condición	Variables hour y minute actualizadas en código con nuevo valor. Campos en Thinger actualizados.

Tabla 47. Prueba2 RF_10

Prueba2 RF_10	
Nombre	Configuración de horario erróneo
Precondición	Campos en Thinger muestran los valores actuales.
Acción	Introducir nueva hora y minuto erróneo (hora: menor de 1 mayor de 24, minuto: menor de 0 mayor de 59) o nulos
Post-condición	Variables hour y minute no actualizadas en código. Campos en Thinger se mantienen.

Tabla 48. Prueba1 RF_11

Prueba1 RF_11	
Nombre	Configuración de umbral
Precondición	Campo en Thinger muestra el valor actual.
Acción	Introducir nuevo valor.
Post-condición	Variable umbral en código con nuevo valor. Campo en Thinger actualizado.

Tabla 49. Prueba2 RF_11

Prueba2 RF_11	
Nombre	Configuración de umbral erróneos
Precondición	Campos en Thinger muestran los valores actuales.
Acción	Introducir valores no válidos. (Por debajo de 0% o por encima del rango máx 25%).
Post-condición	Variables umbral en código no actualizado. Campo en Thinger se mantiene.

Tabla 50. Prueba1 RF_12

Prueba1 RF_12	
Nombre	Historial
Precondición	
Acción	Realizar acción en electroválvula o forzar acción automática
Post-condición	Cambios en Thinger “Data buckets” – electroválvula, donde se muestra la fecha y horario de la acción y su estado.

Tabla 51. Prueba1 RF_13

Prueba1 RF_13	
Nombre	Consulta de datos (BD)
Precondición	
Acción	Registro de valores por los sensores.
Post-condición	Cambios en Thinger "Dashboards" – gráficas, con los valores actualizados y cambios en Thinger "Data buckets" – selección, donde se muestra la fecha y valor registrado.

Tabla 52. Prueba1 RF_14

Prueba1 RF_14	
Nombre	Envío de alertas
Precondición	
Acción	Forzar acción en modo automático y programación.
Post-condición	El usuario recibe el email oportuno con la información del estado del sistema y condiciones climáticas.

4. Gestión del proyecto

4.1. Introducción

En este apartado se trata la organización y gestión del proyecto. Uno de los temas más importantes dentro del desarrollo de un proyecto, puesto que la mayoría de ellos fracasan por una mala planificación y gestión.

Es importante cumplir los plazos establecidos con el cliente y desarrollar un producto mínimo viable (MVP) en paralelo a medida que avanzamos para saber si se van cumpliendo las expectativas del cliente.

4.2. Estimación de recursos

Sección orientada a detallar la estimación de recursos software, hardware y humanos necesarios para completar el proyecto.

4.2.1. Recursos Hardware

El proyecto está planteado para una arquitectura Arduino o microcontrolador compatible, con sensores que recogen información. Para su correcto funcionamiento necesitamos numerosos recursos hardware de todo tipo:

- Un ordenador, no necesariamente de altas prestaciones, pero que permita trabaja de forma fluida con desarrollo de código y entornos de pruebas y conexiones USB e inalámbricas.
- Arduino o microcontrolador, preferiblemente con conectividad a internet, vía cable ethernet o wifi, conexión microUSB para conectar a un ordenador para su programación de manera sencilla y suficientes pines de entrada/salida para las conexiones externas a sensores y otros elementos del circuito.
- Sensores que toman medidas relacionadas con el clima y la tierra, como temperatura, humedad, luminosidad y de lluvia.
- Cables, resistencias y otros materiales (relés, diodos, transistores, transformadores…) para la elaboración de un sistema eléctrico correcto y estable.
- Router o conexión a la red para comunicarnos a través de internet y poder subir los datos e interactuar con el sistema.
- Modulos ZigBee (10) para el envío de información de forma inalámbrica.

4.2.2. Recursos Software

Para el desarrollo del proyecto a nivel Software se utilizarán herramientas Open Source o libres y aplicaciones que, aunque requieran algún tipo de licencia, sean accesibles a los miembros de la Universidad de forma gratuita a través de los convenios con Microsoft Imagine (24).

Se necesitan a parte del Software básico del ordenador, que es nuestra herramienta principal para el desarrollo del proyecto, así como un sistema operativo preferiblemente actualizado (24):

- Herramienta para el desarrollo de documentación para la memoria y manual de usuario.
- Entorno de desarrollo y pruebas para la programación del microcontrolador/Arduino.
- Entorno de desarrollo y pruebas para la programación de los módulos ZigBee.
- Herramienta para la elaboración de una presentación para la exposición del proyecto.

4.2.3. Recursos humanos

El proyecto lo realiza una única persona que es responsable de todas las tareas a llevar a cabo desde su inicio hasta su entrega, y posterior mantenimiento si fuera necesario hasta su retirada. Contará con la ayuda de un tutor asignado al proyecto que actuará como “Product owner” o cliente, para orientar al alumno en la elaboración del proyecto.

4.3. Planificación

En la siguiente tabla se muestra la duración de las tareas realizadas a lo largo del proyecto, así como su duración y fechas de inicio y fin.

Es importante que cada tarea quede completada con éxito para así poder pasar a la siguiente y asegurar el correcto funcionamiento a lo largo de todo el proyecto.

Tabla 53. Tareas del proyecto.

NOMBRE DE TAREA	DURACIÓN	COMIENZO	FIN
Proyecto Gestión Domótica de Riego	**141 días**	**lun 19/12/16**	**mie 02/08/17**
Planteamiento del problema	45 días	lun 19/12/16	vie 17/02/17
Objetivos y Alcance	4 días	sáb 18/02/17	mie 22/02/17
Estado del Arte	9 días	jue 23/02/17	mar 07/03/17
Thinger y sus posibilidades	20 días	mie 08/03/17	lun 03/04/17
Diseño y pruebas funcionales de lectura de sensores	5 días	mar 04/04/17	lun 10/04/17
Análisis de Requisitos. Casos de Uso.	7 días	mar 11/04/17	mar 18/04/17
Gestión de proyecto, recursos y presupuesto	9 días	lun 24/04/17	jue 04/05/17
Diseño y Arquitectura	15 días	vie 12/05/17	jue 01/06/17
Codificación y pruebas, MVP. (Must To Have)	15 días	lun 12/06/17	vie 30/06/17
Desarrollo y pruebas MVP. (Nice to have)	5 días	lun 10/07/17	vie 14/07/17
Conclusiones y futuras líneas de investigación	7 días	mar 25/07/17	mie 02/08/17

A continuación, se muestra el Diagrama de Gantt correspondiente, reflejando gráficamente los tiempos empleados para el desarrollo de cada una de las actividades de forma independiente y mostrando una forma más visual el periodo de desarrollo de todo el proyecto.

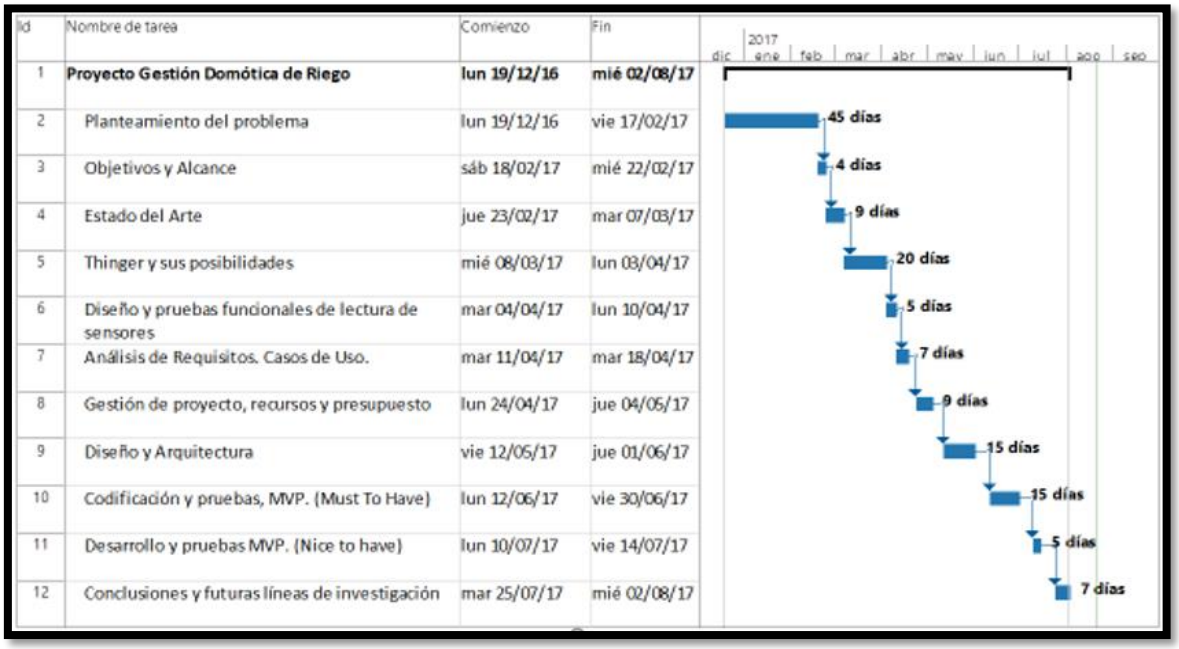


Figura 23. Diagrama de GANTT

4.4. Elección de recursos

A partir de las estimaciones de los recursos necesarios para elaborar el proyecto y el estudio de mercado de las distintas posibilidades que se podrían utilizar para el desarrollo del mismo, seleccionamos los recursos haciendo un esfuerzo en proporcionar la máxima calidad posible en todas las partes sin descuidar el coste final del proyecto, ajustándonos a las funcionalidades y requisitos del sistema.

4.4.1. Material de desarrollo

En cuanto al material necesario para el desarrollo del proyecto se intenta seleccionar el material que nos permita construir el sistema con todas las funcionalidades y una buena relación calidad precio.

Microcontrolador

Como motor para el sistema se utiliza NodeMCU ESP826 1.0 Amica (23). Esta placa fue diseñada principalmente para trabajar con Lua, aunque es posible utilizarla con Micropython o con Arduino. Por lo que al poder trabajar como si fuera una placa Arduino tradicional (19), que lleva integrada conectividad wifi y que su precio es bastante más asequible, nos decantamos por esta placa para el desarrollo de nuestro proyecto.

Especificaciones técnicas de NodeMCU 1.0 (Amica/Lolin) (23).

- Voltaje de entrada (USB): 5V
- Voltaje de salida en los pines: 3.3V
- Voltaje de referencia en el ADC: 3.3V
- Corriente nominal por pin: 12mA
- Frecuencia de procesador: 80MHz (160MHz max.)
- 4MB Flash
- Consumo de corriente en stand-by @80MHz: 80mA

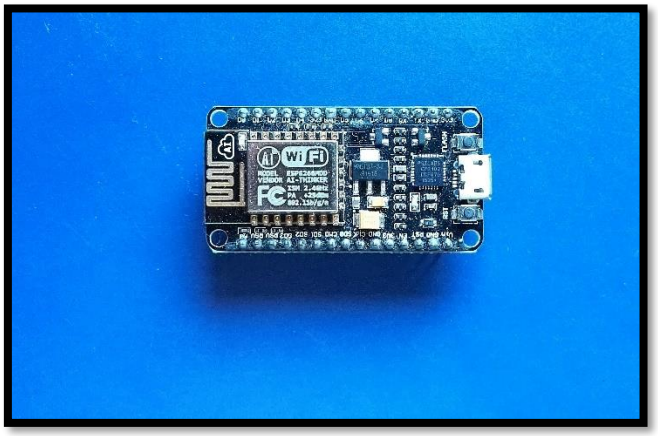


Figura 24. Placa Node MCU 1.0 Amica

Sensor temperatura y humedad

Utilizaremos un sensor DHT22 (6) ya que, para las necesidades de nuestro objetivo, es una calidad más que suficiente a buen precio, con una instalación sencilla y gran precisión. Además, cuenta con una fotorresistencia LDR GL5537 para obtener datos de luminosidad ambiental (6).

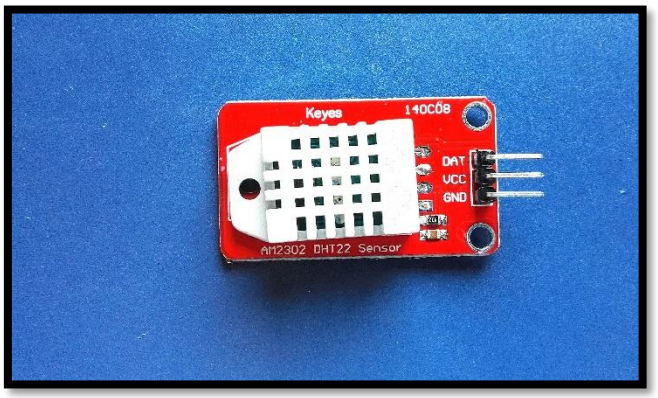


Figura 25. Sensor DHT22

Sensor LDR

Utilizaremos una fotorresistencia GL5537 para recoger la información luminosa del ambiente, debemos conectar este sensor a la placa a través de otra resistencia para regular la caída de voltaje y no quemar el sensor (25).

Figura 26. Sensor LDR

Sensor de humedad de la tierra

Utilizaremos el sensor YL-69 junto con el controlador YL-38 con salida analógica, para medir la humedad del suelo, que se obtiene a partir de la conductividad de las puntas que varía en función de la humedad del suelo. Estará alojado en el módulo ZigBee aislado en uno de los pines analógicos y se mandará su información a través de la tecnología ZigBee hasta la placa NodeMCU (7).

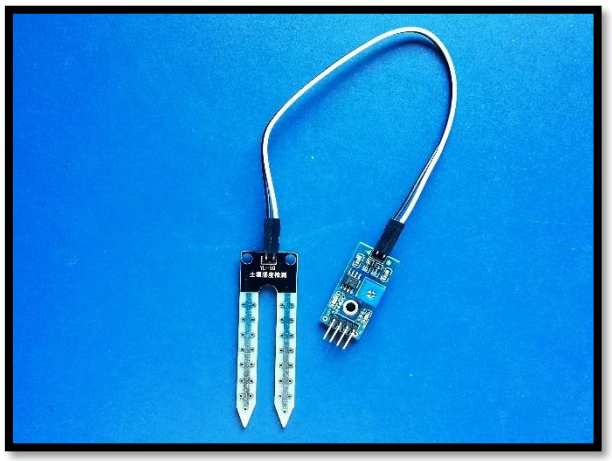


Figura 27. Sensor de humedad de la tierra

Sensor de lluvia

Utilizaremos un sensor YL-83 junto con el controlador YL-38 con salida digital, para detectar si está lloviendo en un momento dado, para evitar regar innecesariamente cuando llueve (8).

Figura 28. Sensor de lluvia

Bluetooth Vs ZigBee

Bluetooth está pensado para aplicaciones como los teléfonos móviles o la informática casera y tiene un consumo eléctrico relativamente alto en comparación con ZigBee. De hecho, en términos exactos, Bluetooth tiene un consumo de 40mA transmitiendo y 0.2mA en reposo, frente a los 30mA transmitiendo y 3µA en reposo de ZigBee (9).

Es por esta razón por la que Bluetooth no es una elección óptima, pues este requiere de un bajo consumo ya que va a depender de una pequeña batería.

ZigBee parece una tecnología ideal (10). Presenta un bajo consumo, una tasa de envío de datos reducida (aunque más que suficiente para nuestros propósitos), un alcance de más de 100 metros y lo más importante, es una tecnología creada para la interconexión de pequeños dispositivos con batería en una red de múltiples nodos.

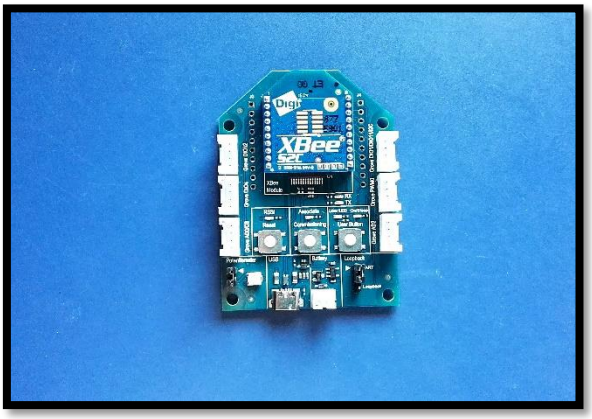


Figura 29. Módulo ZigBee

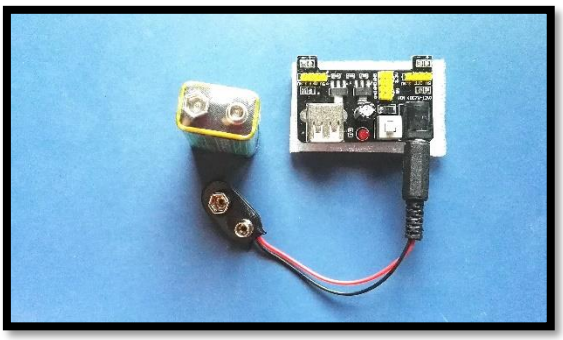


Figura 30. Tranformador. Pila 9V - 5V o 3V

Elección de la alimentación.

Una vez analizadas las ventajas y desventajas de cada tipo de batería se pueden concluir con que, para una aplicación como la nuestra para el módulo ZigBee aislado, en la cual las temperaturas no van a ser extremas, la mejor elección es una batería alcalina típica de 9V con un transformador a 5V, porque ofrece una buena calidad precio.

Nuestro modulo principal formado por la placa y la electroválvula deben estar suministrados preferiblemente por corriente física del interior del hogar o instalación cercana, con el transformador de 24V para la electroválvula y uno de 5V para la placa (típica de móvil).

Al tratarse de un sistema de prueba el tamaño de la pila, no requiere gran importancia, sin embargo, de cara a un posible sistema embebido, habría que emplear una pila alcalina de botón que pueda ser integrada fácilmente. También se podría considerar adaptarla a un pequeño panel solar para su recarga durante las horas de Sol, de esta manera la pila duraría mucho más tiempo, dependiendo del lugar de instalación.

Figura 31. Tranformador a 24V

Figura 32. Tranformador a 5V

Relé

Se utilizará un único relé con una resistencia de 10ª y 25V, tiene integrado una resistencia y un diodo, para estabilizar la corriente y que esta no circule hacia atrás pudiendo dañar nuestra placa. Al lado de entrada irá conectada la placa con la señal de control de la electroválvula y por el lado de salida estará conectada la electroválvula y el transformador de corriente de 24V para hacerla funcionar.

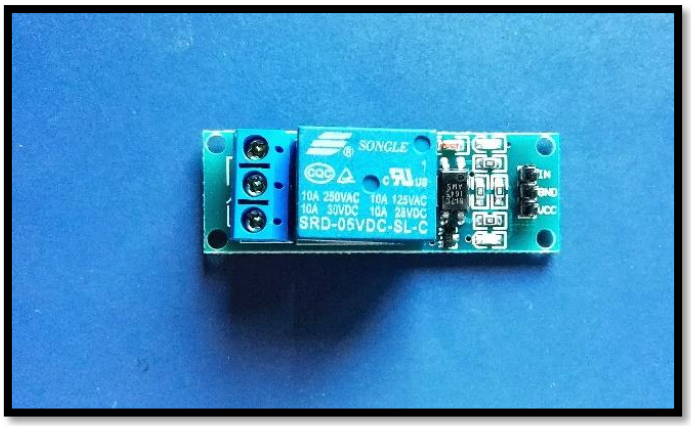


Figura 33.Relé

Electroválvula

La electroválvula elegida es Rain Bird 100-HV (12). Más fiable, versátil, fácil de instalar y con menor mantenimiento. Tiene un diseño compacto con cuerpo en polipropileno reforzado con fibra de vidrio para una mayor resistencia y durabilidad.

Especificaciones:

- Presión: 1,0 a 10,3 bares.
- Caudal: 0,05 a 6,82 m3/h.
- Temperatura: temperatura máxima del agua de 43°C; temperatura ambiente máxima de 52°C.

Especificaciones eléctricas

- Electroválvula de 24 v de CA a 50/60 Hz.
- Corriente de entrada máxima: 0,250 Amperios a 60 Hz.
- Corriente de retención: 0,143 Amperios a 60 Hz.
- Resistencia de la bobina: 52 a 55 Ohmios.

Figura 34. Electroválvula

4.4.2. Herramientas de desarrollo

Las herramientas que se han utilizado a lo largo de la elaboración del proyecto a nivel Software han sido:

Microsoft Windows 10

Windows 10 es la versión más actualizada de los Sistemas Operativos de Microsoft. Es el Sistema Operativo elegido en el área de trabajo (26).

Figura 35. Windows 10

Microsoft Word 2016

Microsoft Word 2016 es un Software orientado al procesamiento de textos. Este software ha sido diseñado por la empresa Microsoft (27).

Figura 36. Word 2016

Microsoft PowerPoint 2016

Microsoft PowerPoint 2016 es el nombre de uno de los programas más populares creados por Microsoft. Se trata de un software que permite realizar presentaciones a través de diapositivas. El programa contempla la posibilidad de utilizar texto, imágenes, música y animaciones (28).

Figura 37. PowerPoint 2016

Microsoft Visio 2016

Microsoft Visio 2016 es un software de dibujo vectorial para Microsoft Windows. Las herramientas que lo componen permiten realizar diagramas de oficinas, diagramas de bases de datos, diagramas de flujo de programas, UML, y más, que permiten iniciar al usuario en los lenguajes de programación (29).

Figura 38. Wicrosoft Visio 2016

Project profesional

Project profesional (o MSP) es una herramienta de Microsoft para administración de proyectos diseñado, desarrollado y comercializado por Microsoft para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo (30).

Figura 39. Project profesional

Thinger.io

Plataforma libre para trabajar con internet de las cosas (IoT). La consola web (Cloud Console) permite controlar, configurar y administrar tus dispositivos y visualizar la información en la nube (20).

Figura 40. Thinger.io

XCTU

XCTU es una aplicación multi-plataforma gratuita diseñada para permitir a los desarrolladores interactuar con los módulos RF de Digi a través de una interfaz gráfica sencilla de usar. Incluye herramientas que facilitan la configuración y prueba de los módulos XBee® RF (21).

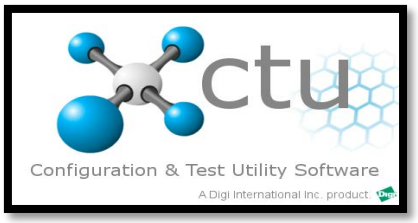


Figura 41. CTU

4.5. Presupuesto

A partir de los requisitos y de los recursos disponibles y necesarios, se realiza un presupuesto completo del proyecto a desarrollar.

4.5.1. Costes directos

Los costes directos son aquellos relacionados directamente con el proyecto o los servicios que ofrece. En este caso los recursos que suponen coste son en su mayoría material hardware que se empleará para desarrollar adecuadamente el sistema.

Personal

En este proyecto solo está implicada una persona, que está a cargo de todas las partes en su totalidad. Sin embargo, se trata de un proyecto que se lleva a cabo de manera alterna en el tiempo y gestionado por la propia persona. Por lo tanto, calculando una media de 5 horas empleadas por día de trabajo, por los 141 días efectivos dedicados al proyecto.

Considerando un sueldo de 1.500€ al mes por un Ingeniero Junior que trabaja 8 horas al día. Son 1.500/30 = 50€ el día trabajado / 8 = 6.25€ la hora. Por lo tanto, 6,25 * 5 * 141 = 4.406,25 € el coste bruto total del personal, con tipo de cotización del 20%.

Tabla 54. Costes personal

Recurso	Hombre/mes	Coste bruto	Seg. Social	Coste total
Ingeniero JR	1.500,00 €	4.406,25 €	881,25 €	5.287,50 €

Equipo

El equipo utilizado para el desarrollo del proyecto supone otro de los costes principales. Sin embargo, no se trata de una compra exclusiva para el desarrollo de este sistema, por lo tanto, se realizan los cálculos a partir de una amortización de uso estimada en 4 años.

Tabla 55. Costes equipo

Recurso	Coste	Uso	Amortización	Coste real
PC portátil	700,00 €	5 meses	48 meses	72,92 €

Hardware

Aquí incorporamos todos los costes relativos al sistema físico del proyecto.

Tabla 56. Costes Hardware

Recurso	Unidades	Coste real
NodeMCU 1.0 Amica	1 unidad	72,92 €
Electroválvula es Rain Bird 100-HV	1 unidad	19,75 €
Transformador 24V	1 unidad	19,90 €
Sensor DHT11	1 unidad	2,24 €
Sensor DHT22	1 unidad	7,99 €
Sensor LDR	5 unidades	1,75 €
Sensor lluvia YL-83	1 unidad	1,64 €
Sensor humedad de la tierra YL-69	5 unidades	6,69 €
Relé	5 unidades	8,99 €
Kit cables, resistencias, leds y otros	1 unidad	19,90 €
Módulos Xbee	2 unidades	50,00 €
TOTAL		146,84 €

Software

Los costes que en cuanto a Software necesario son nulos gracias a los convenios de la universidad con las licencias de las distintas herramientas, y otro software de acceso libre.

Tabla 57. Costes Software

Descripción	Unidades	Coste
Windows 10	1 unidad	0,00 €
Windows Word 2016	1 unidad	0,00 €
Windows PowerPoint 2016	1 unidad	0,00 €
Windows Excel 2016	1 unidad	0,00 €
Windows Project 2013	1 unidad	0,00 €
Windows Visio 2016	1 unidad	0,00 €
Arduino IDE	1 unidad	0,00 €
Xcode	1 unidad	0,00 €
TOTAL		**0,00 €**

4.5.2. Costes indirectos

Los costes indirectos en este proyecto corresponden únicamente a los conocimientos empleados durante el desarrollo del mismo y a otras actividades relacionadas como transporte de mercancías, desplazamiento a compra de materiales, y otras tareas de administración y finanzas. Se estiman en un 15% del total de los costes directos.

Tabla 58. Costes Indirectos

	Recurso	Coste
Costes directos	Ingeniero	5.287,50 €
	PC portátil	72,92 €
	Hardware	146,84 €
	Software	0,00 €
	TOTAL	5.507,26 €
Costes indirectos (15%)		826,09 €

4.5.3. Costes totales

En la siguiente sección se muestra la agrupación de los costes totales implicados en la totalidad del proyecto.

Tabla 59. Costes totales

Recurso	Coste
Costes directos	5.507,26 €
Costes indirectos	826,09 €
Coste total de proyecto	**6.333,35 €**

Al total del proyecto, añadimos un margen de riesgo del 15%, por posibles inconvenientes que puedan surgir a lo largo del desarrollo. También se añade un 15% como margen de beneficio, como desarrollo y venta del producto final.

Tabla 60. Coste final.

Recurso	Coste
Coste total del proyecto	6.333,35 €
Margen de riesgo 15%	950,00 €
Coste total + margen de riesgo	7.283,35 €
Margen de beneficio 15%	1.092,50 €
TOTAL	**8.375,85 €**

5. Diseño, arquitectura e implementación

En este apartado se trata de esclarecer el diseño global del producto, tanto software como hardware y la comunicación entre todas las partes, así como la implementación desarrollada.

5.1. Arquitectura

Para el desarrollo de este proyecto, se ha considerado aplicar un patrón de arquitectura software basado en: Modelo – Vista – Controlador (MVC). Esta arquitectura, se basa en las ideas de reutilización de código y la separación de conceptos, influyendo positivamente en la facilidad de desarrollo y mantenimiento.

- **Modelo**: Es la capa donde se trabaja con los datos, contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos habitualmente en una base de datos. En nuestro caso, está formado por los datos almacenados en los “Data Buckets”, que son enviados por el controlador.
- **Vista**: Contiene el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario. El usuario interactúa con las distintas vistas para solicitar información o ejecutar acciones en el sistema de riego.
- **Controlador**: Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades del sistema. El controlador, es el encargado de mandar la información recogida por los sensores, y de la activación física de la electroválvula.

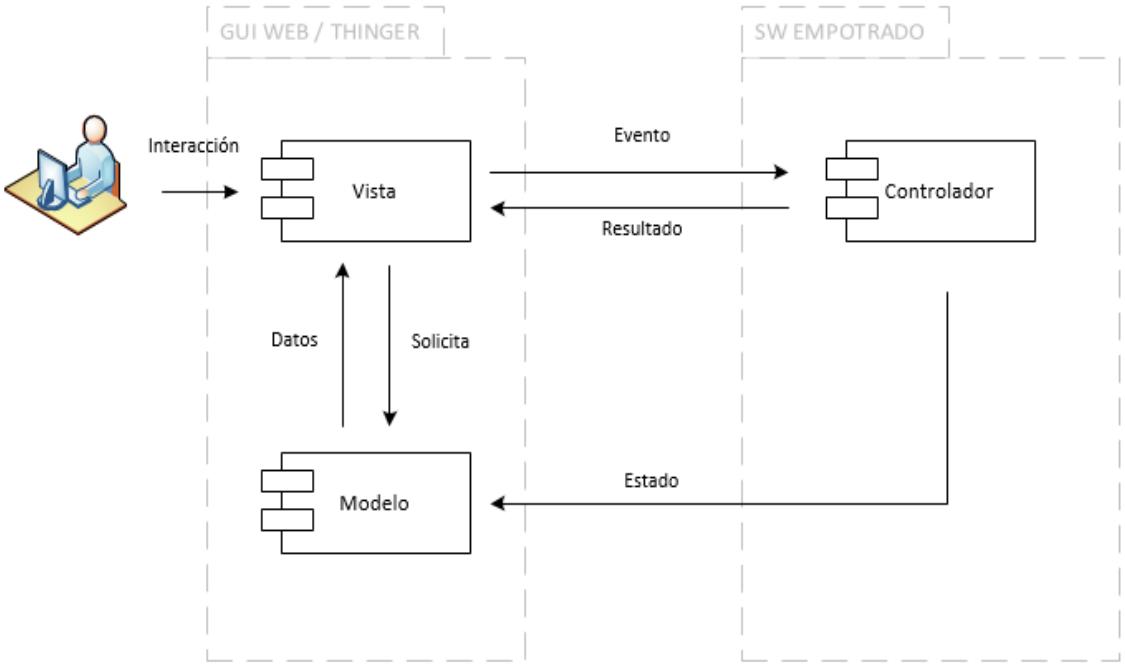


Figura 42. Modelo - Vista - Controlador

5.2. Vista lógica

En esta sección se describe la arquitectura que se desea construir, mediante el uso de un diagrama de clases basadas en los requisitos detallados anteriormente.

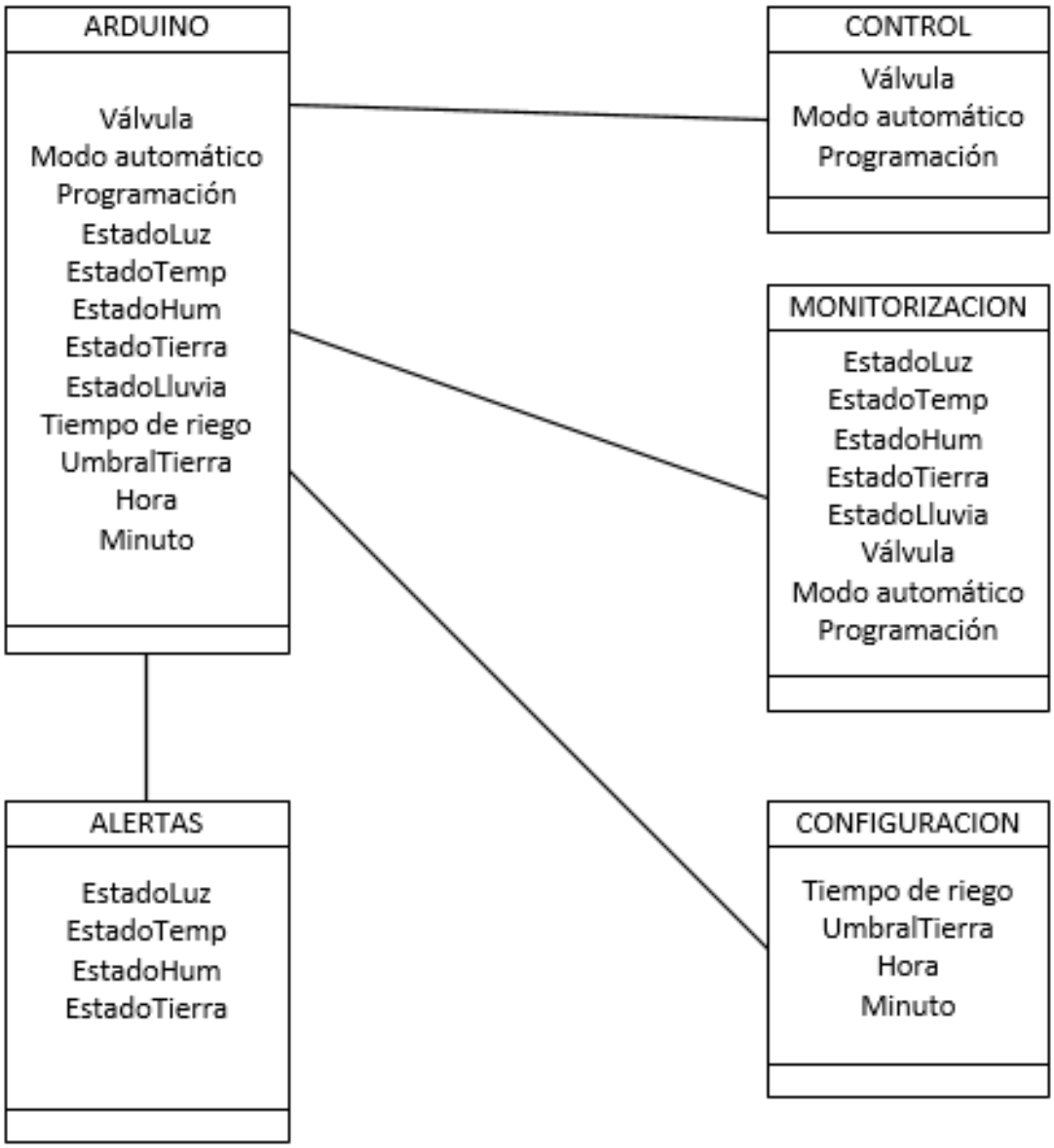


Figura 43. Arquitectura. Imagen propia de archivo

El diagrama representa la información que usarán los componentes con las dependencias que tienen entre sí.

Para el control de riego, el Arduino (software de la placa NodeMCU), utiliza la información de los umbrales establecidos, los estados a tiempo real de la electroválvula y los modos de automatización, además del tiempo de riego y los valores que se recogerá mediante los sensores.

Para la monitorización, la interfaz mostrará al usuario el estado de la electroválvula, los valores medidos por los sensores y el modo de funcionamiento que está activado en ese momento bien sea automático o manual, y si la programación a una hora concreta está activada.

Para la configuración se manejarán los umbrales de encendido y apagado del riego dependiendo de los valores del entorno, y también la configuración de la programación a partir de la hora y minuto.

Por defecto se establecen los umbrales nulos (cero), y es labor del usuario decidir dichos valores para el modo automático. El usuario podrá establecer los umbrales en un rango establecido en los requisitos.

Por último, se representa el control que tendrá el estado de la electroválvula, el modo de funcionamiento que está activo en ese momento y si la programación esta activada.

5.3. Vista de desarrollo

El componente Arduino se encargará de realizar la ejecución del sistema de riego ya sea en modo manual, automático o programado. Es el software integrado en la placa NodeMCU (23) y, se encargará de la comunicación con la plataforma, recogerá los valores de los sensores y actuará sobre las electroválvulas cuando sea necesario o se indique desde el SW (interfaz web/ Thinger) (20) alojado en internet (servidor web).

El componente Control que se encargará de indicarle a Arduino el modo de funcionamiento activos, así como de realizar el control manual de riego (activar / desactivar electroválvula).

El componente monitorización que se encargará de mostrar en la interfaz gráfica (data buckets y dashboards) los datos que reciba del componente Arduino.

El componente Configuración que será el encargado de permitir configurar los umbrales de los sensores para el riego en el modo automático, el tiempo de riego y la hora y minuto del modo programación.

El componente Alertas se encargará de avisar al usuario vía correo electrónico cuando se active de forma automática la electroválvula por acción del modo automático o por la programación.

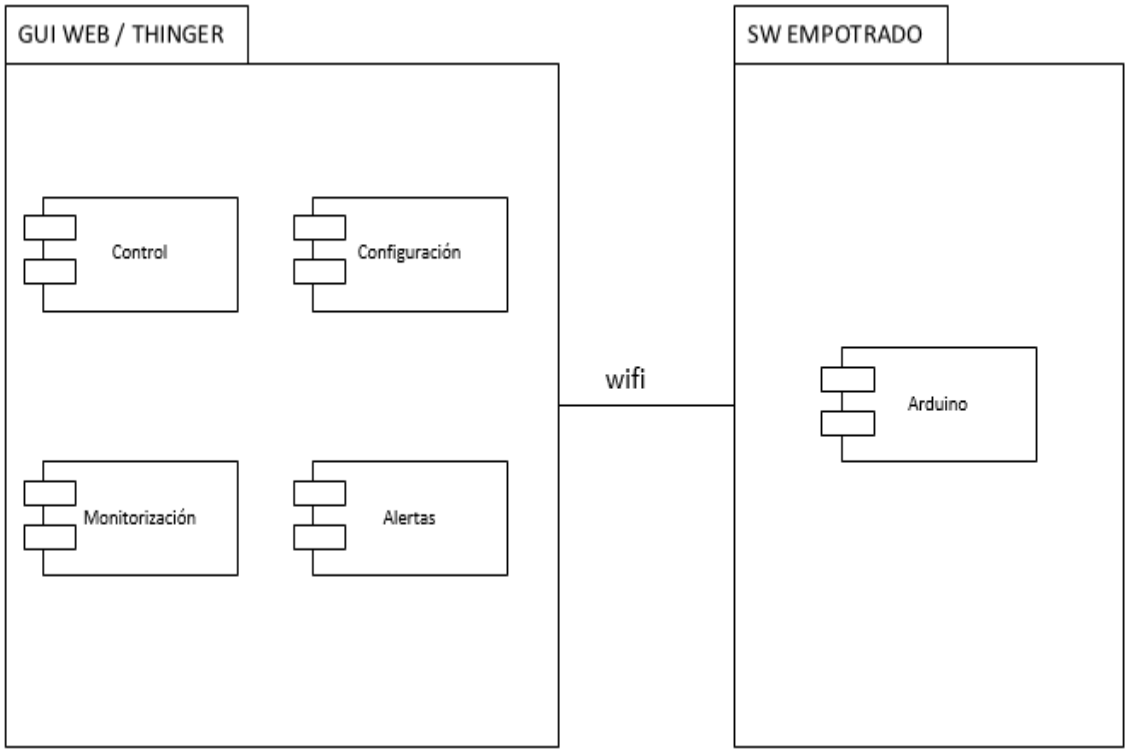


Figura 44.Vista de desarrollo. Imagen propia de archivo

5.4. Vista de procesos

En este apartado se detalla con un diagrama de estados cómo los objetos de nuestro sistema modifican su estado en función de eventos, sucesos que ocurren en el sistema, o el paso del tiempo.

- Estado: representa un determinado periodo de tiempo del sistema, durante el que se produce una acción o características determinadas y espera un evento o estímulo que cambie su estado.
- Evento: representa una operación, estímulo o suceso que hace cambiar el estado del sistema.
- Elección: representa una operación o comprobación tras un evento, que determina el estado siguiente.

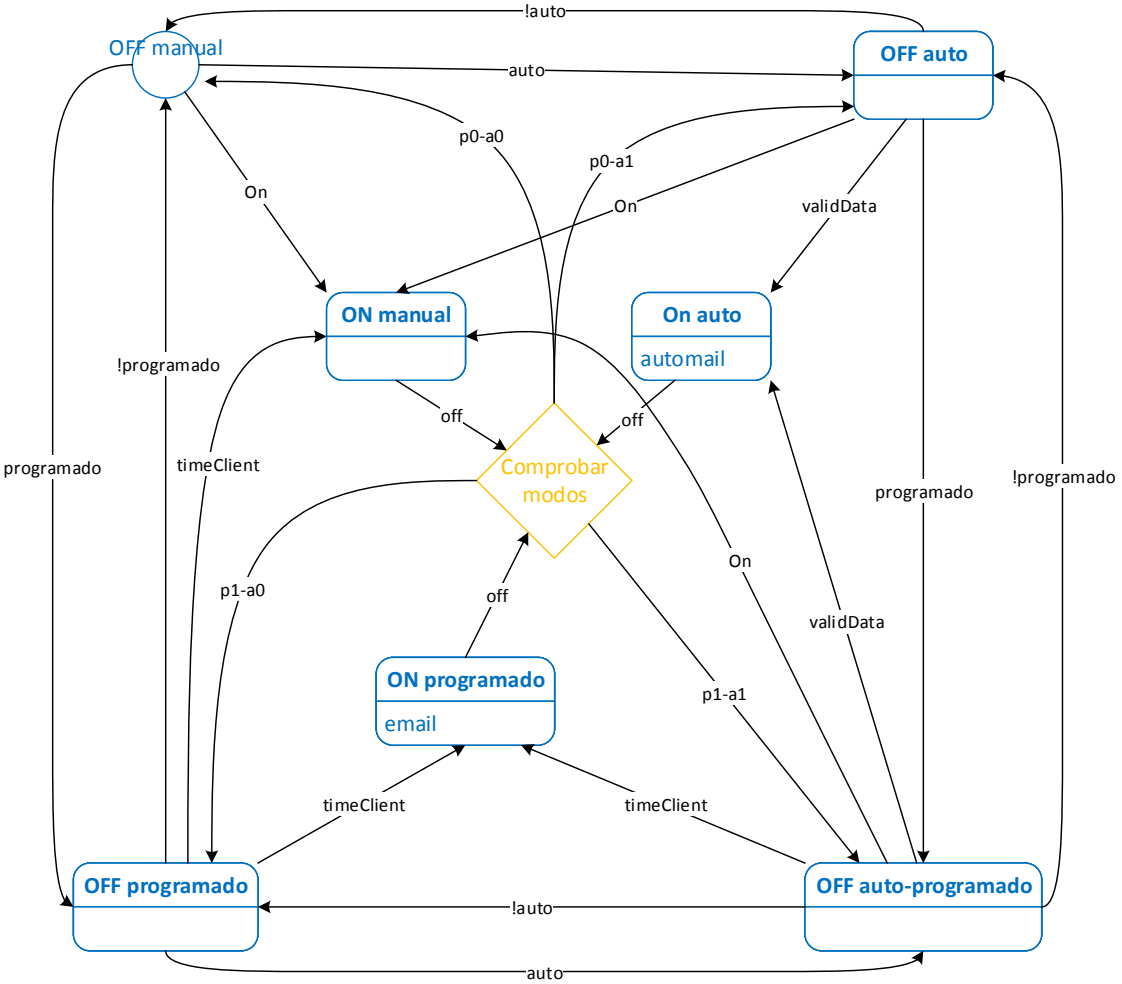
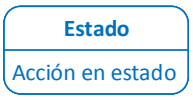


Figura 45. Diagrama de Estados.

Leyenda:

-  Estado inicial
- 

Estado
-  Elección
-  Evento

Interpretación del diagrama:

- Estados
  - OFF manual: representa el estado inicial del sistema, donde los modos automático y programado están desactivados y la electroválvula cerrada.
  - OFF auto: representa el estado en el que únicamente el modo automático está activado y la electroválvula cerrada.
  - OFF programado: representa el estado en el que únicamente el modo programación está activado y la electroválvula cerrada.
  - OFF auto-programado: representa el estado en el que el modo automático y programación están activados y la electroválvula cerrada.
  - ON manual: estado en el que la electroválvula está abierta por acción manual del usuario.
  - ON auto: estado en el que la electroválvula está abierta por el cumplimiento de las condiciones propias de la activación automática. Al entrar en el estado, se efectúa el envío de un email al usuario con parámetros climáticos.
  - ON programado: estado en el que la electroválvula está abierta por el cumplimiento del horario establecido por el usuario. Al entrar en el estado, se efectúa el envío de un email al usuario con parámetros climáticos.
- Eventos:
  - programado: acción del usuario al activar el modo programación.
  - auto: acción del usuario al activar el modo automático.
  - !programado: acción del usuario al desactivar el modo programación.
  - !auto: acción del usuario al desactivar el modo automático.
  - On: corresponde a la acción manual por parte del usuario de activar la electroválvula.

- Off: corresponde a la desactivación de la electroválvula, ya sea por acción manual del usuario, o por cumplimiento del tiempo de riego.
- timeClient: evento correspondiente a la coincidencia de la hora y minuto establecidos por el usuario con la actual y cumplimiento de la restricción de este modo (humedadDeLaTierra<60%).
- validData: evento correspondiente al cumplimiento de las condiciones y restricciones del modo automático (humedadDeLaTierra < umbral, horario, temperatura y luminosidad).
- p0-a0: modo automático y programación desactivados.
- p0-a1: modo automático activado y modo programación desactivado.
- P1-a0: modo automático desactivado y modo programación activado.
- P1-a1: modo automático y programación activados.

- Elección:
  - Comprobar modos: Al finalizar el riego o activación de la electroválvula ya sea de forma manual o por finalización del tiempo de riego. Se comprueban que modos están activos, para determinar el próximo estado.

5.5. Diseño de circuitos (Vista física)

Módulo central (coordinador)

Para medir la luminosidad ambiental, se crea un sencillo divisor de tensión, para ellos se utiliza una resistencia de 1KΩ y un LDR conectado a los 3V se salida de la placa, de esta manera la corriente circulará a través del LDR cuya resistencia oscilará en función de la luz que incida sobre él. Se conecta la salida del LDR a un pin analógico de la placa (A0) para evaluar la medida del sensor.

Para comprobar el estado de las precipitaciones, se utiliza el módulo YL-38 conectado al sensor YL-83. El sensor YL-83 transmite la diferencia de potencial medida por sus pistas conductoras. El módulo controlador YL-38, previamente calibrado mediante su potenciómetro integrado, compara el voltaje y manda una señal de salida digital (0/1) según corresponda al pin de la placa (D3).

Se utiliza el DHT-22 para las lecturas de temperatura y humedad ambiental, está integrado en un módulo con resistencias y un led de encendido, por lo que sus conexiones son muy sencillas. Se conecta la salida de datos del sensor, al pin digital (D7) de la placa, necesitamos una librería especial para proceder a la lectura de estos datos digitales.

Para el control de la electroválvula se utiliza un relé integrado en un módulo con resistencias y otros limitadores de corriente como diodo y transistor para evitar que la corriente circule hacia atrás y se queme el circuito. Está conectado a la placa, y a través del pin (D5) de la placa se controla la activación de su interruptor, que cierra el circuito de su otro extremo y activa la electroválvula, localizada en la salida “normalmente cerrada”, alimentada por un transformador de 24V.

Para la comunicación entre el microcontrolador y el módulo XBee coordinador, se utiliza la comunicación serial. Los módulos XBee funcionan a 3.3V y los pines no son tolerantes a 5V. Desde el microcontrolador podemos alimentar un módulo XBee, pero la comunicación serie es a 5V y en el módulo XBee es a 3.3V. Por ello, usamos un divisor de tensión.

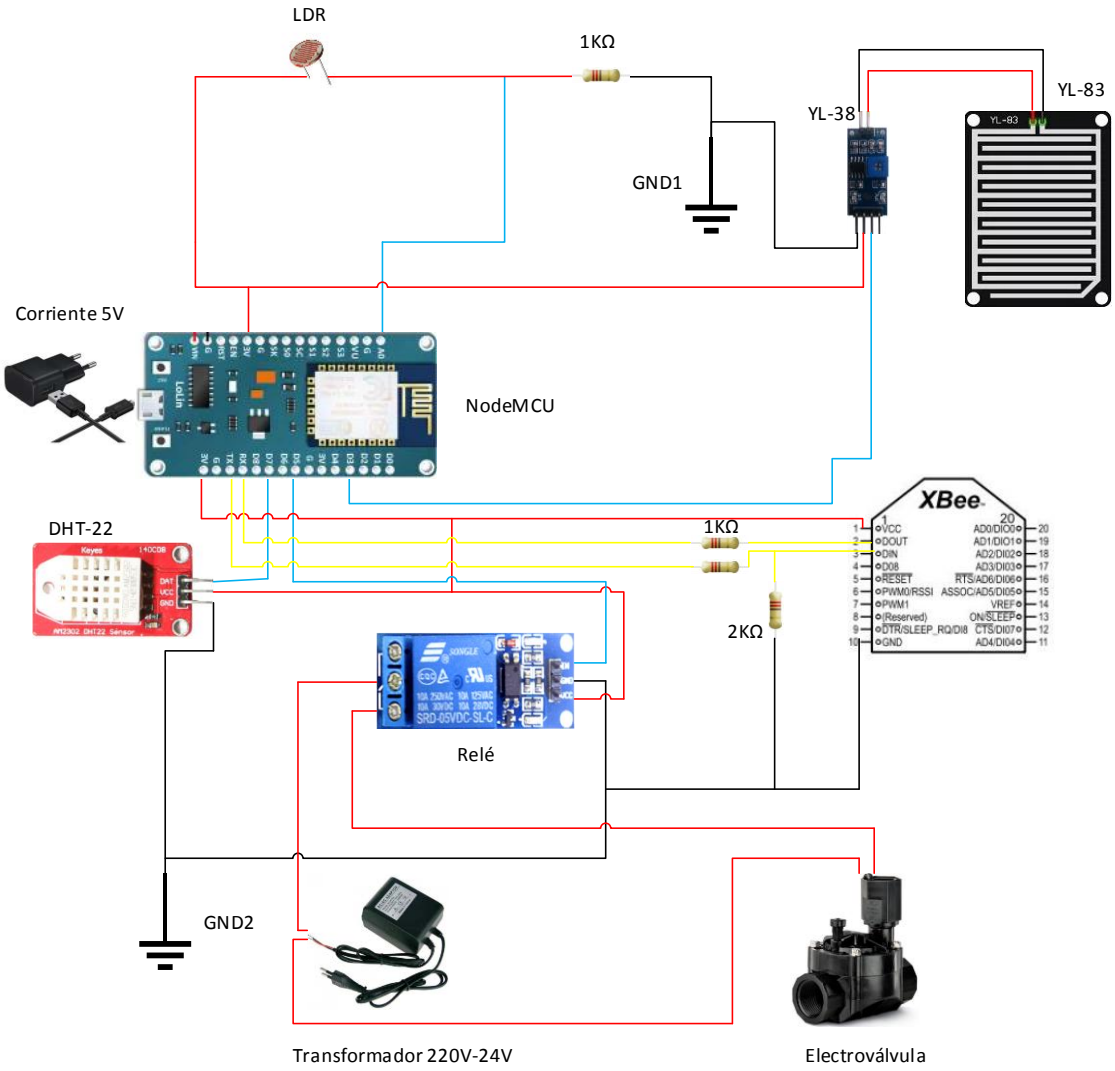


Figura 46. Arquitectura del sistema.

Figura 47. Arquitectura física del sistema.

Módulo final (dispositivo final)

Para comprobar la humedad del suelo, utilizamos el módulo YL-38 conectado al sensor YL-69. El sensor YL-69 transmite la diferencia de potencial medida por sus puntas conductoras, el módulo controlador YL-38 compara el voltaje y manda una señal de salida analógica (0-1023) al pin (A0) del módulo XBee.

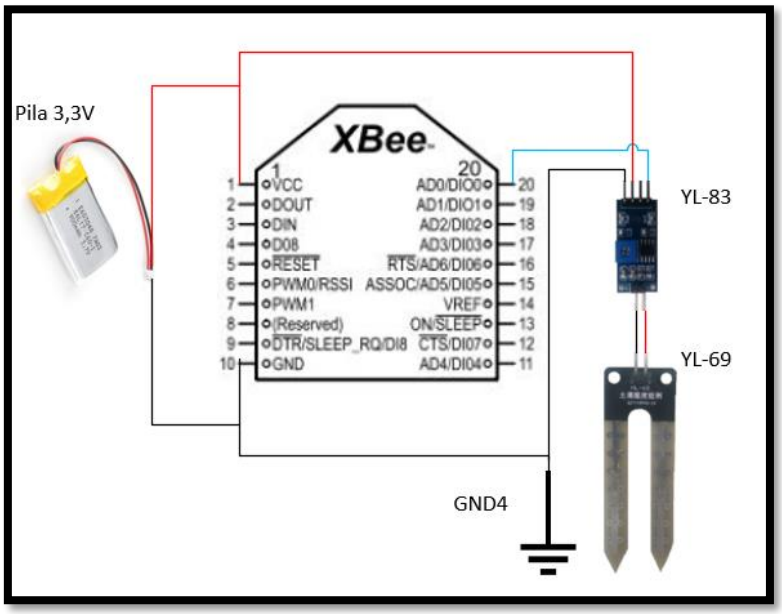


Figura 48. Módulo final.

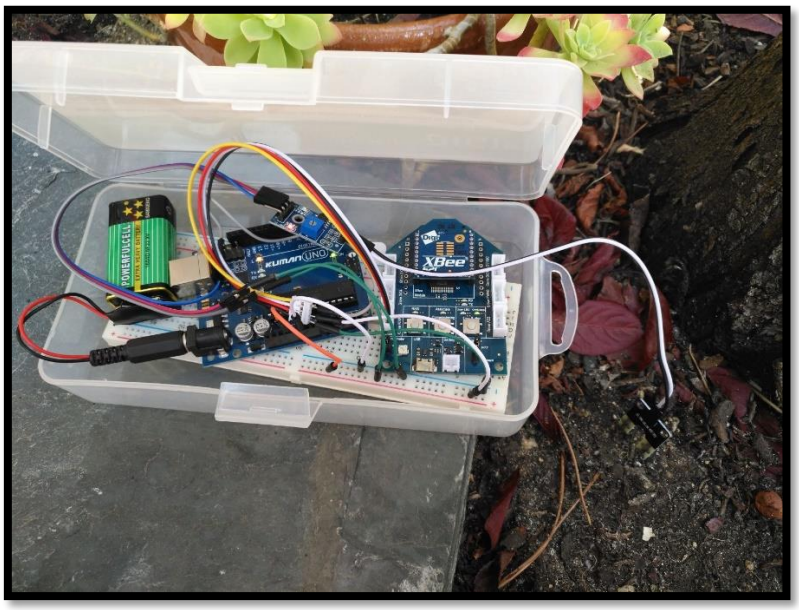


Figura 49. Módulo físico final .

5.6. Diseño de Arquitectura ZigBee

Una red XBee la forman básicamente 3 tipos de elementos. Un único dispositivo Coordinador, dispositivos Routers y dispositivos finales (end points). Los módulos XBee son versátiles a la hora de establecer diversas topologías de red, dependiendo la serie de XBee que escojamos pueden crearse redes (31).

El Coordinador: Es el nodo de la red que tiene la única función de formar una red. Es el responsable de establecer el canal de comunicaciones y del PAN ID (identificador de red) para toda la red. Una vez establecidos estos parámetros, el Coordinador puede formar una red, permitiendo unirse a él dispositivos Routers y End Points. Una vez formada la red, el Coordinador hace las funciones de Router, esto es, participar en el enrutado de paquetes y ser origen y/o destinatario de información.

Los Routers: Es un nodo que crea y mantiene información sobre la red para determinar la mejor ruta para enrutar un paquete de información. Lógicamente un Router debe unirse a una red Zigbee antes de poder actuar como Router retransmitiendo paquetes de otros Routers o de End Points.

End Device: Los dispositivos finales no tienen capacidad de enrutar paquetes. Deben interactuar siempre a través de su nodo padre, ya sea este un Coordinador o un Router, es decir, no puede enviar información directamente a otro End Device. Normalmente estos equipos van alimentados a baterías. El consumo es menor al no tener que realizar funciones de enrutamiento y poder permanecer en hibernación.

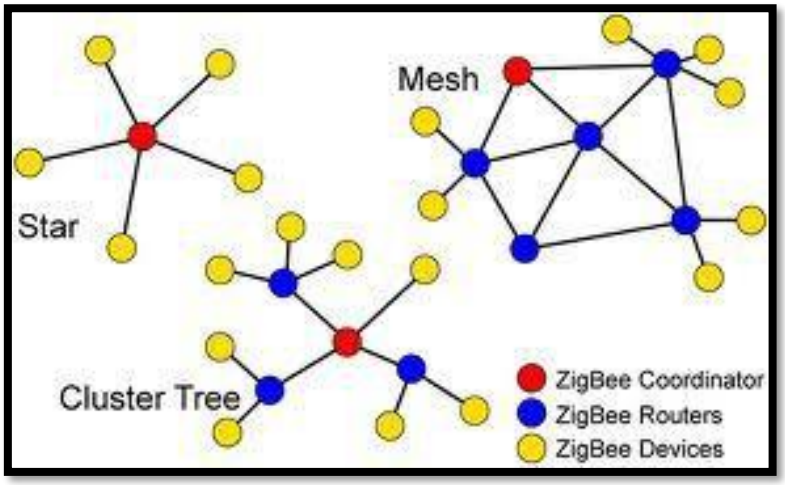


Figura 50. Tipos de arquitectura ZigBee.

Para nuestro sistema implementamos una arquitectura en forma de estrella (star), puesto que se trata de una integración sencilla y escalable donde únicamente es necesario un módulo central conectado al microcontrolador que será el módulo XBee coordinador, y uno o varios dispositivos finales para recoger información de determinados terrenos o jardines. De esta manera, podemos realizar la monitorización de distintas extensiones de terreno desde un mismo microcontrolador.

5.7. Diseño y desarrollo sobre Plataforma Thinger.io

En este apartado entraremos en detalle en la especificaciones y funcionalidades que nos permite la plataforma Thinger (20) y cómo utilizar, diseñar y configurar para hacer cumplir nuestros requisitos y funcionalidades del sistema.

En primer lugar, se deberá crear una cuenta con un correo electrónico para poder usar los servicios que oferta, existen varias opciones acordes a las restricciones de éstos. Para el desarrollo de este proyecto es suficiente con la versión gratuita.

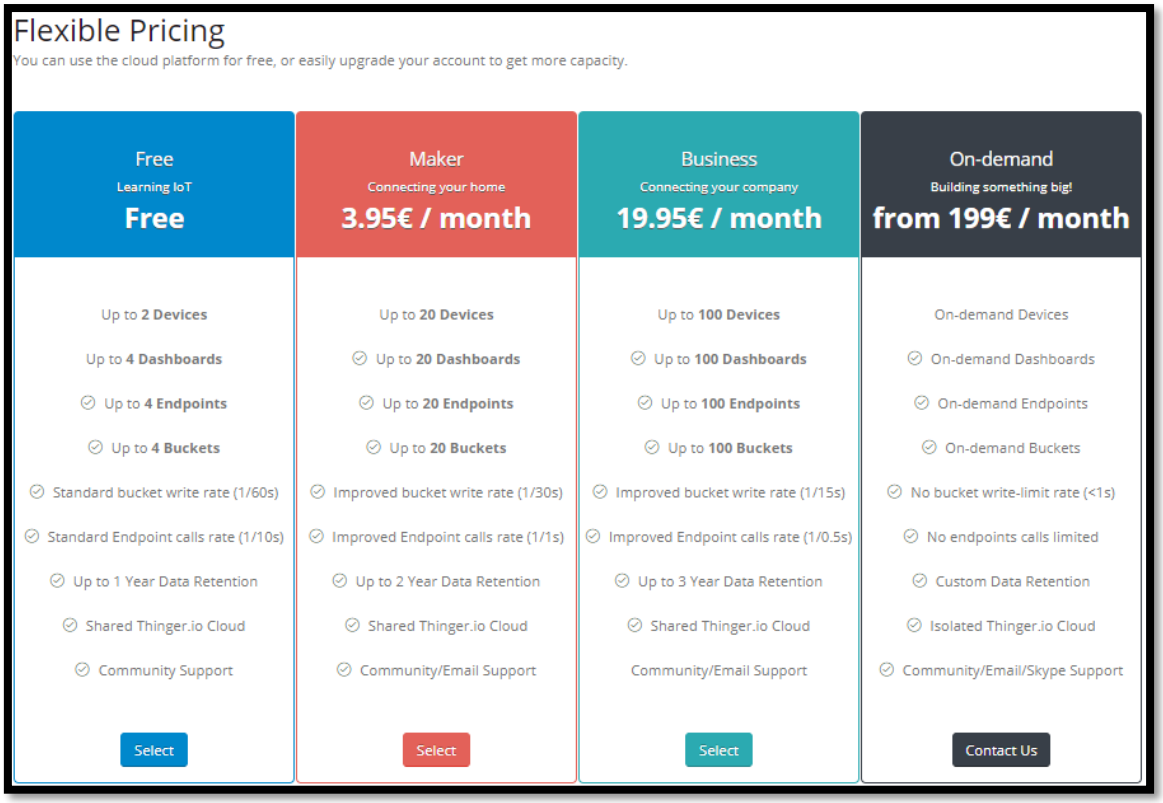


Figura 51. Flexible Pricing

Una vez creada la cuenta y logueados en la plataforma, tenemos un menú vertical en el margen izquierdo por el que navegar.

En “Statistics” tenemos un resumen visual de los recursos disponibles y utilizados, que hemos definido.

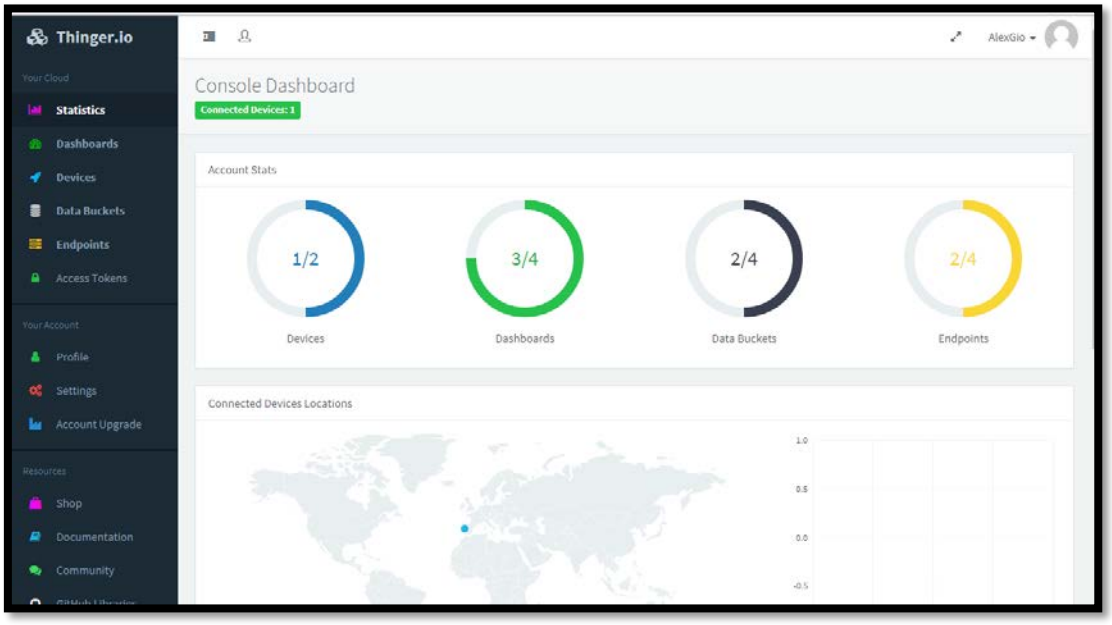


Figura 52. Console Dashboard Thinger

En “Dashboards”, se pueden crear y administrar nuestras visualizaciones, en nuestro caso se han diseñado 3 interfaces, asociadas a los componentes de configuración, monitorización y control.

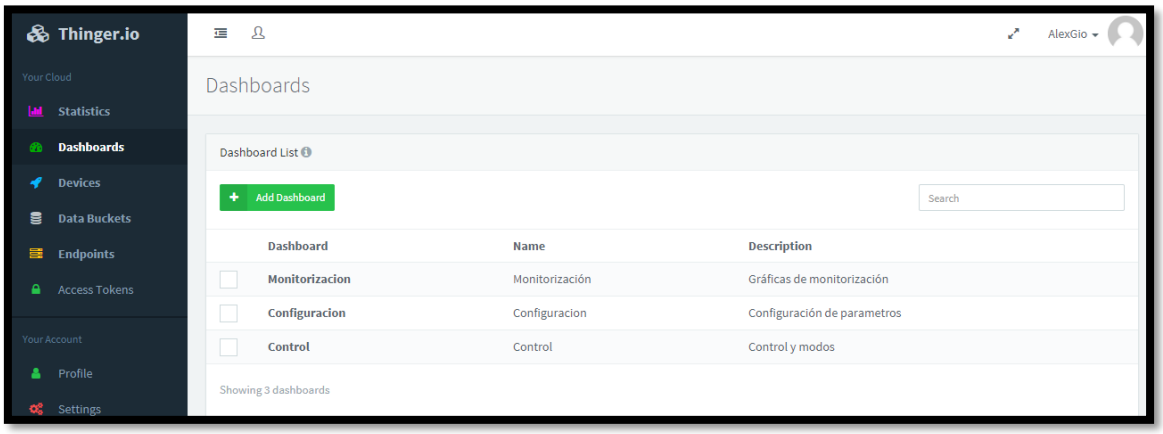


Figura 53. Dashboards Thinger

En el de configuración, se han diseñado unas barras numéricas para poder establecer la hora y el minuto del riego programado, el tiempo de riego y el umbral de humedad de la tierra.

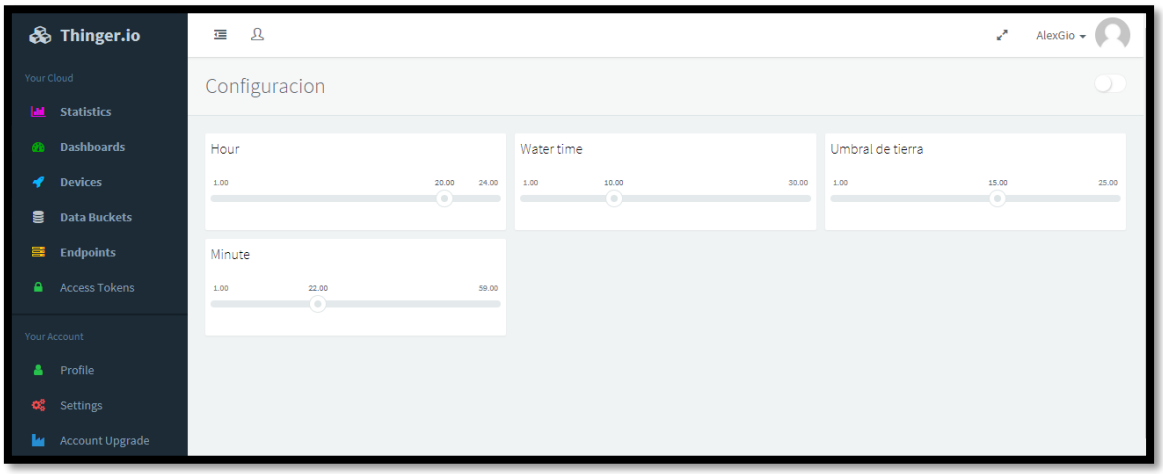


Figura 54. Configuración Thinger

En monitorización, se han diseñado unas graficas asociadas a los valores recogidos por los sensores y guardados en los Data Buckets. De esta manera, podemos visualizar los datos de forma sencilla y se percibe una idea cuantitativa del estado climatológico y de su evolución.

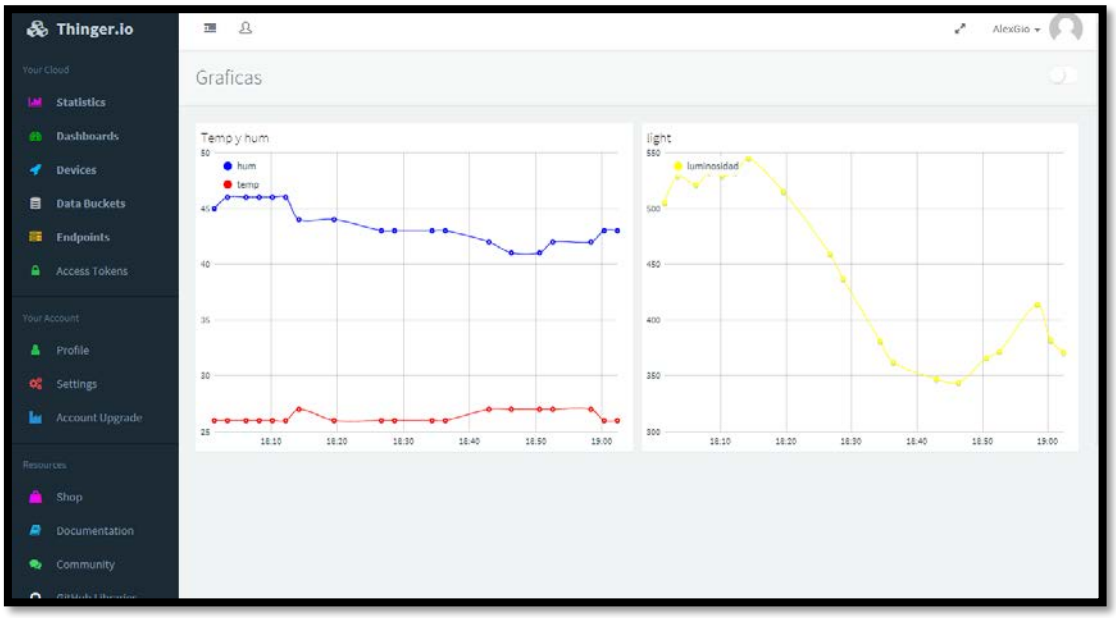


Figura 55. Graficas Thinger

En control, se dispone de unos botones accionables que determinan la activación y desactivación de la electroválvula y el modo automático y programación.

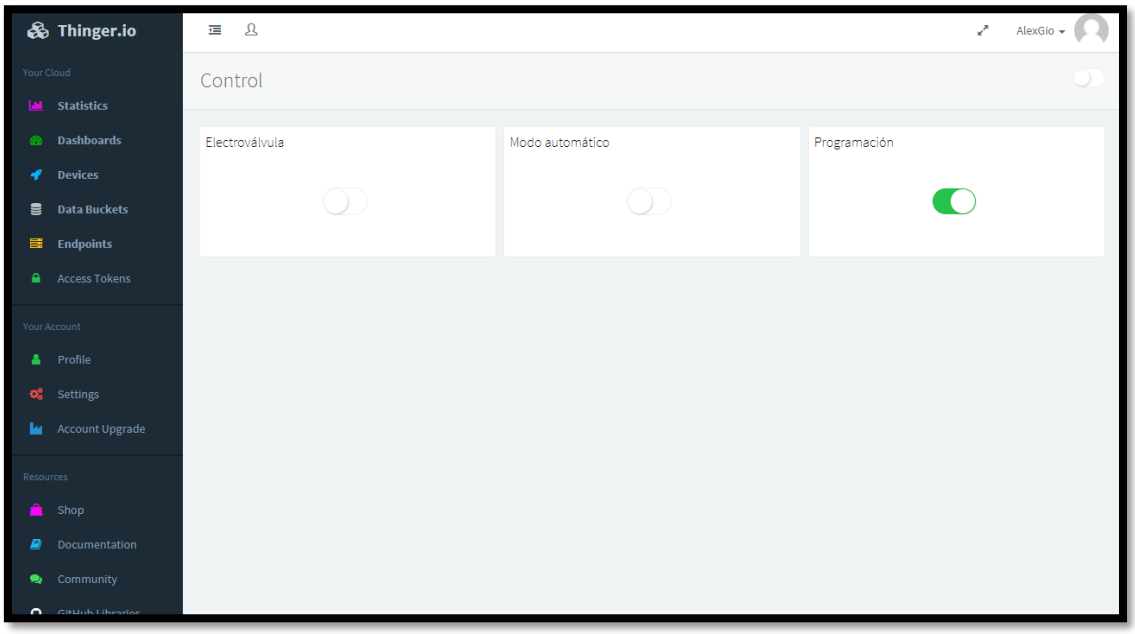


Figura 56. Control Thinger

Desde “Devices”, podemos configurar nuevos dispositivos y administrarlos. Se muestra una lista con su estado y seleccionando accedemos a su API de funcionalidades.

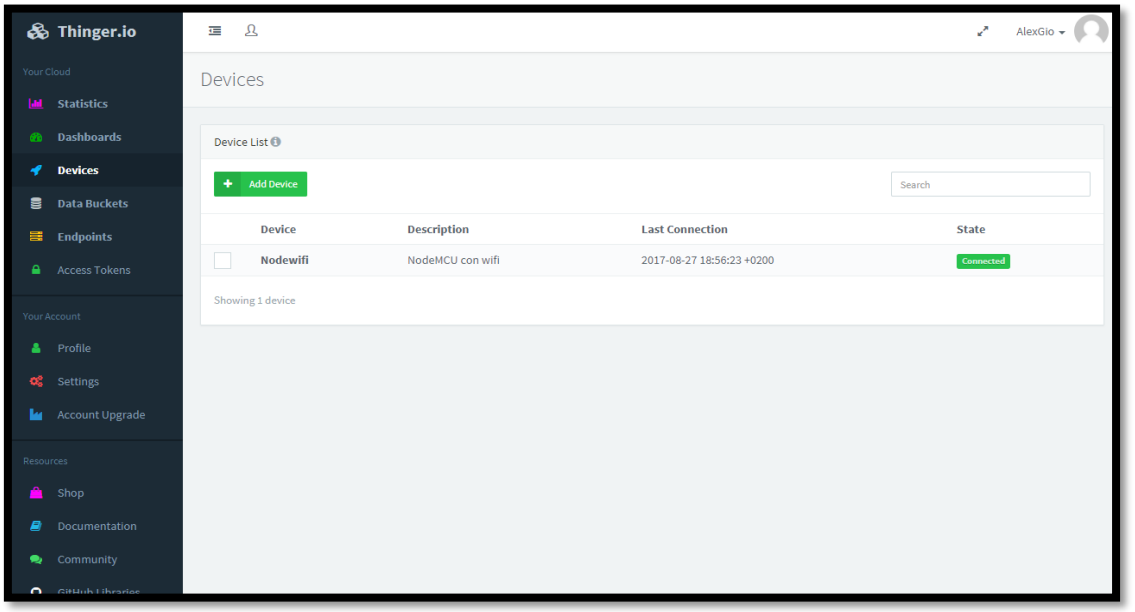


Figura 57. Devices 2 Thinger

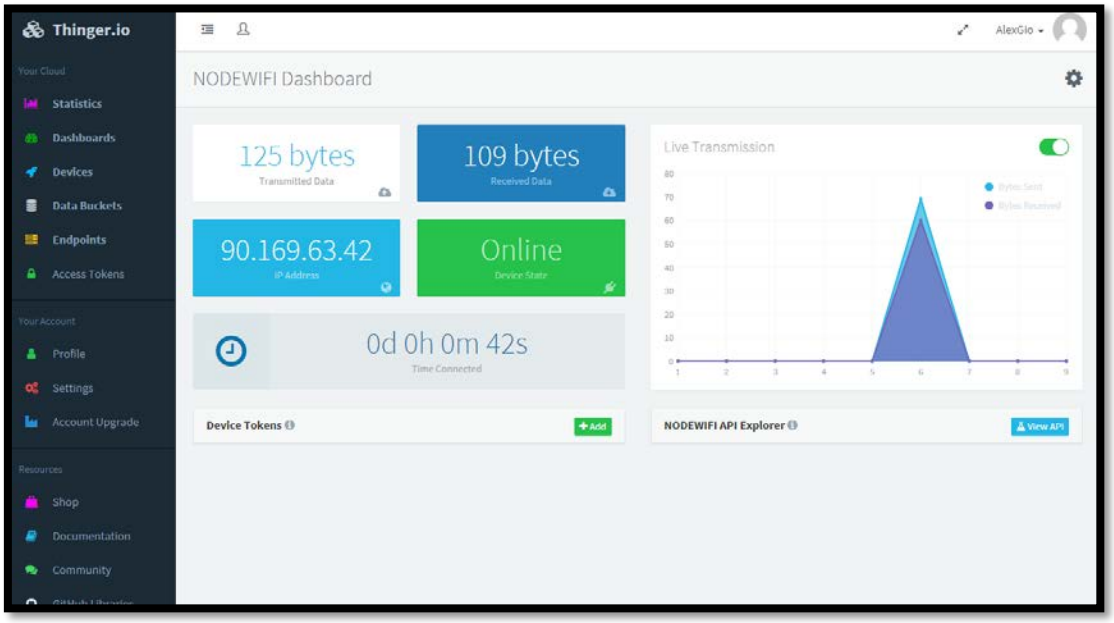


Figura 58. Nodewifi Dashboard Thinger

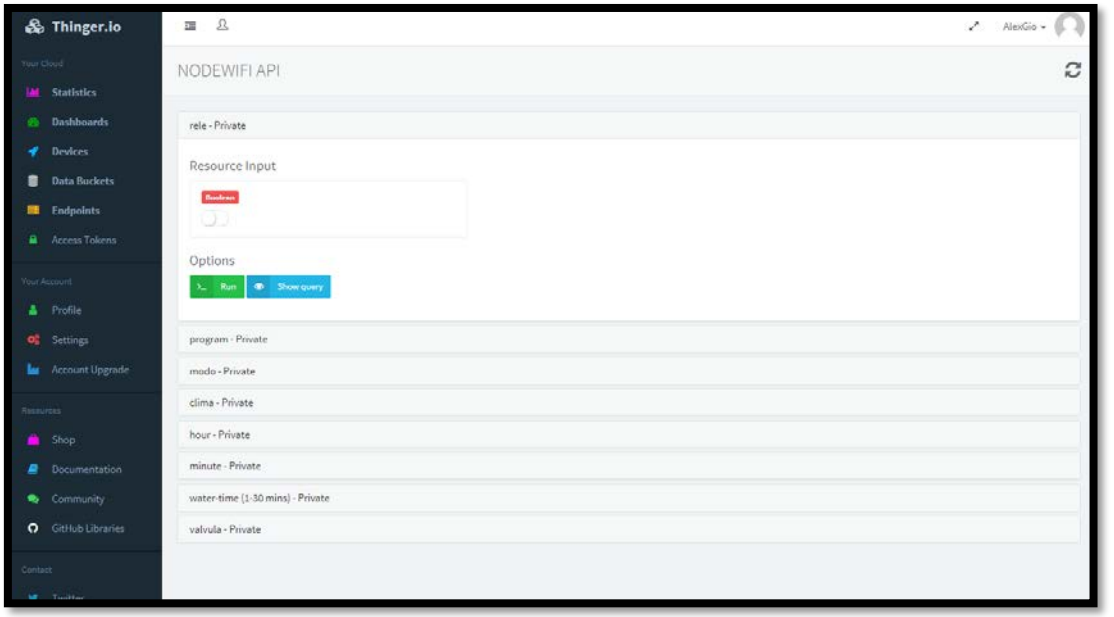


Figura 59. Nodewifi API Thinger

En “Data Buckets”, se gestionan los datos registrados por el sistema (placa y sensores) y subidos a la plataforma.

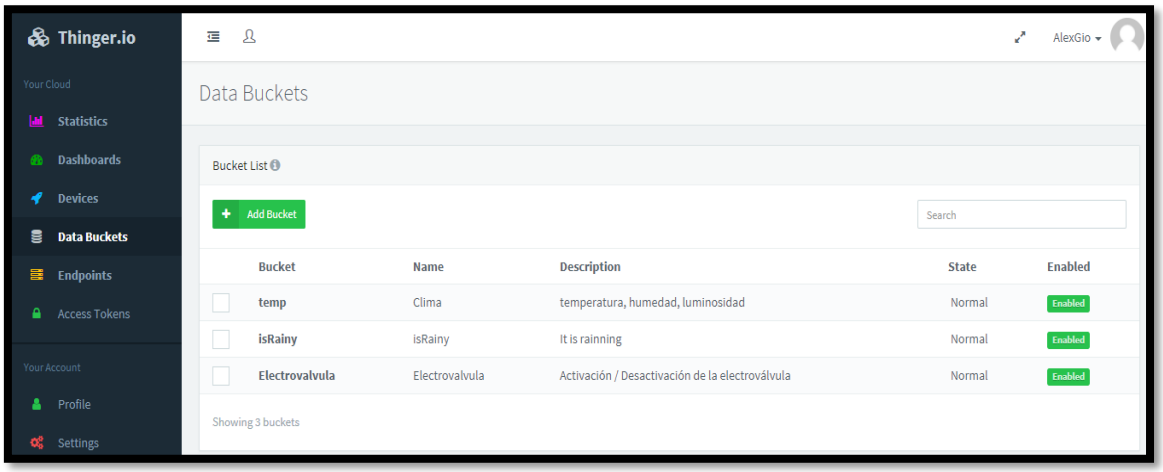


Figura 60. Data Buckets Thinger

Aquí se muestra un ejemplo de la información que manda la placa al producirse un cambio en el estado de la electroválvula.

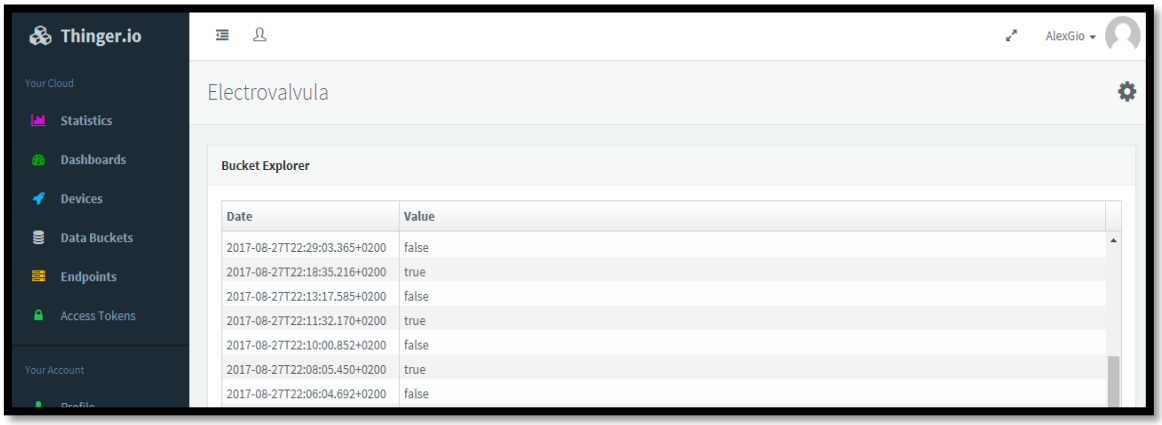


Figura 61. Estado Electroválvula Thinger

Desde aquí, también podemos administrar los datos, exportarlos o eliminarlos.

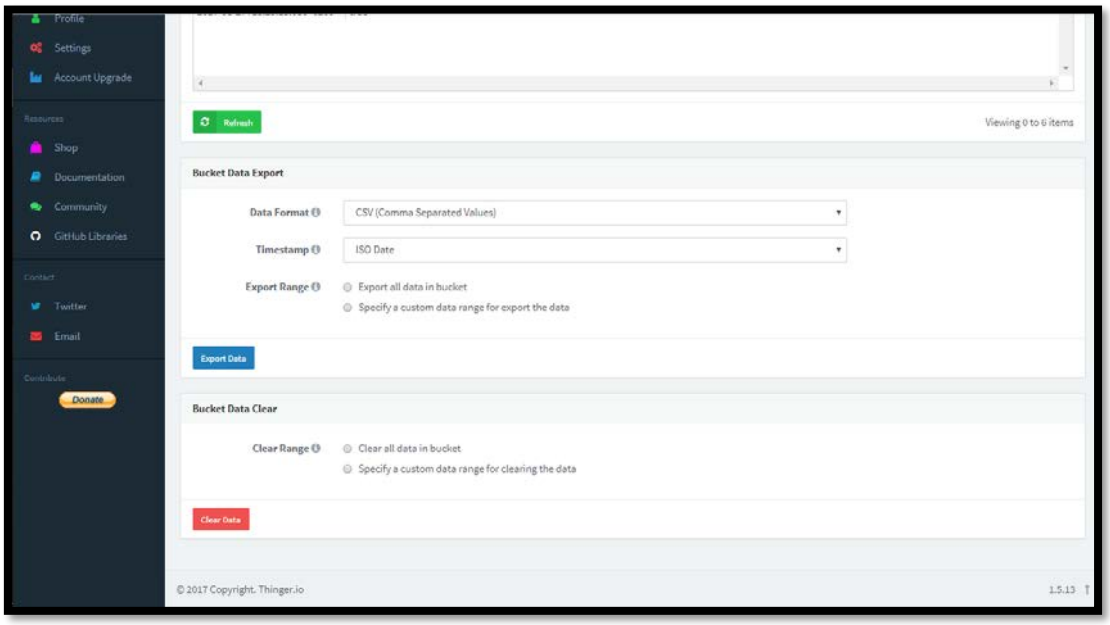


Figura 62. Válvula 2 Thinger

Aquí se muestra otro ejemplo con la información registrada por los sensores. En este caso la información se envía cada cierto tiempo, fijado dependiendo de cada cuanto queramos disponer de la información ambiental.

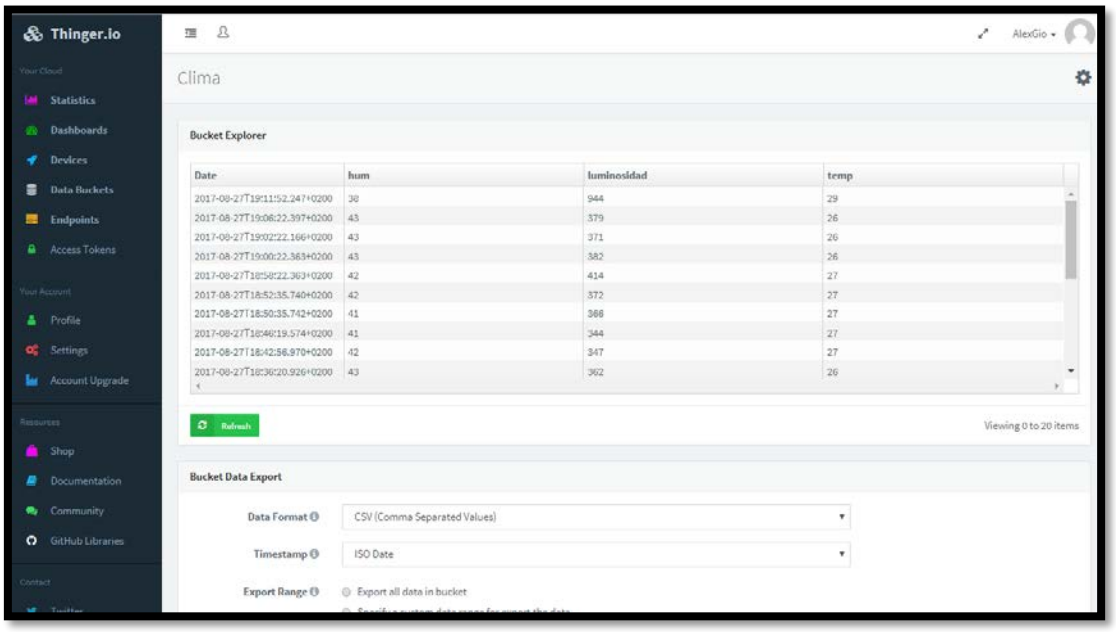


Figura 63. Clima Thinger

En “Endpoints”, podemos configurar una serie de servicios (email, mensaje, llamada…) en forma de alertas o avisos, de esta manera podemos controlar algún fallo o situación concreta. En nuestro caso, se han programado 2 envíos de correo electrónico en caso de activación automática de la electroválvula por acción del modo automático o programación.

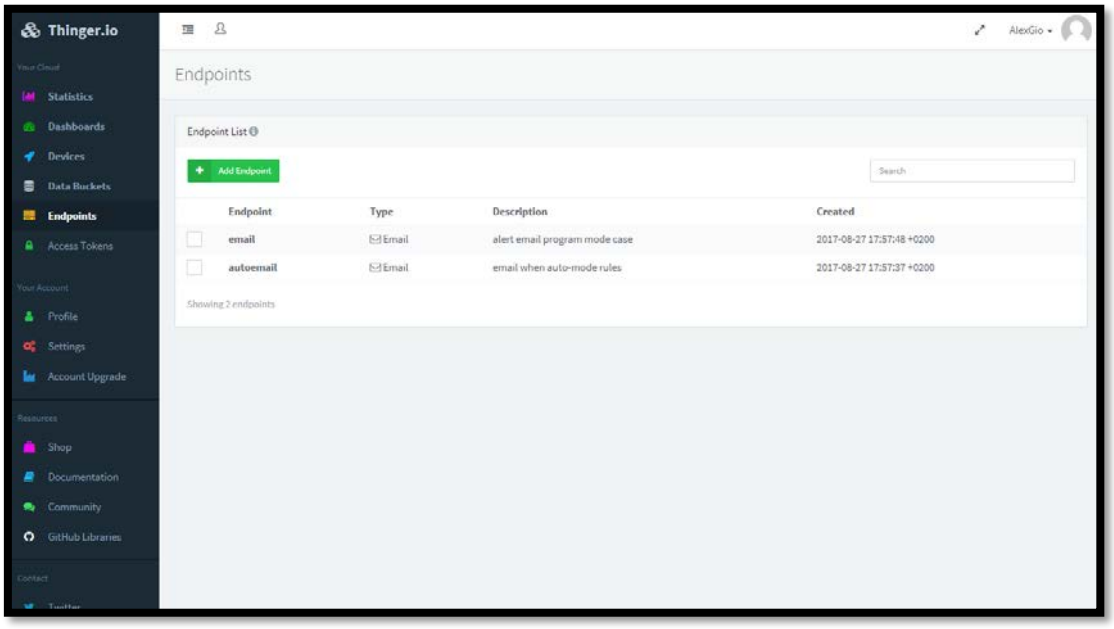


Figura 64. Endpoints Thinger

Configuración desde Thinger, destinatario y asunto.

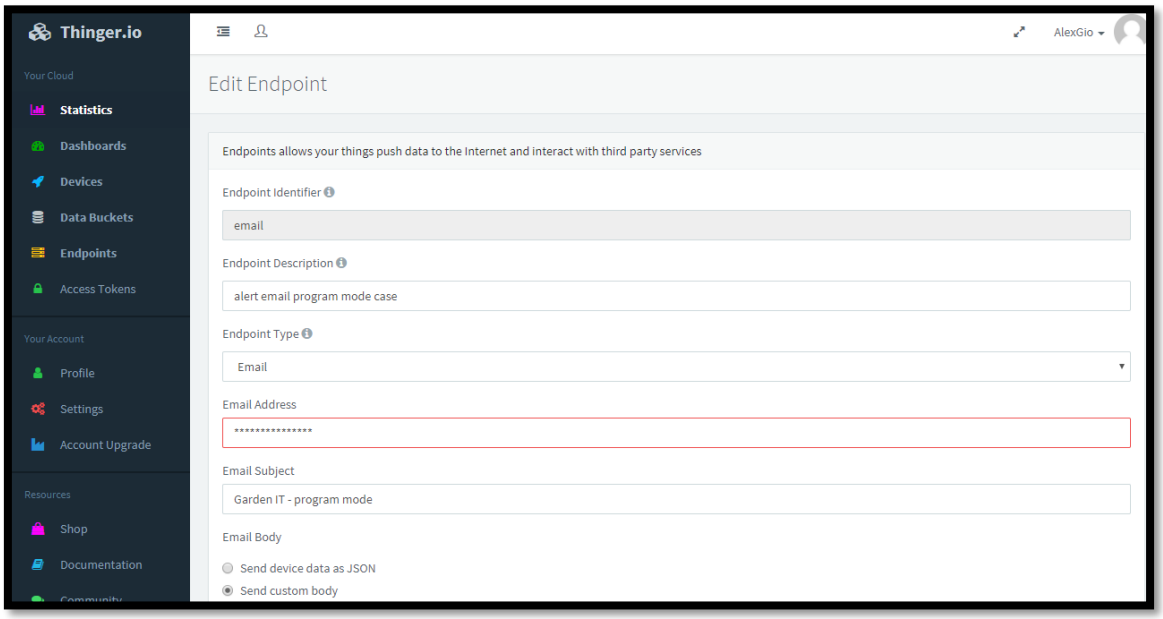


Figura 65. Edit Endpoint Thinger

Configuración desde Thinger del cuerpo del correo electrónico, donde declaramos los parámetros a enviar.

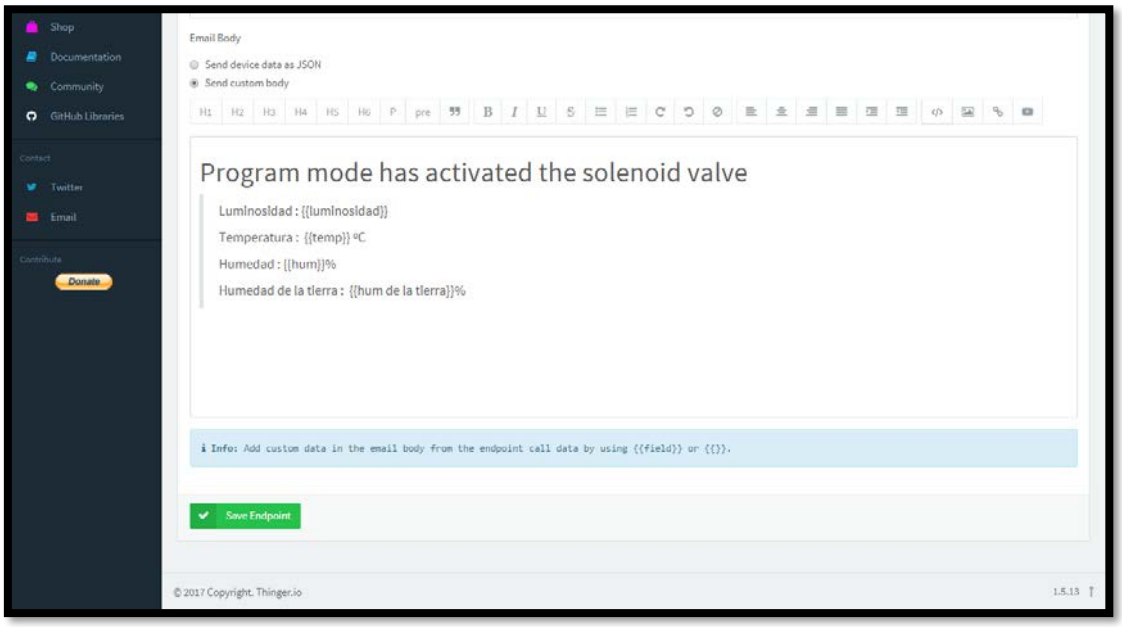


Figura 66. Endpoints 3 Thinger

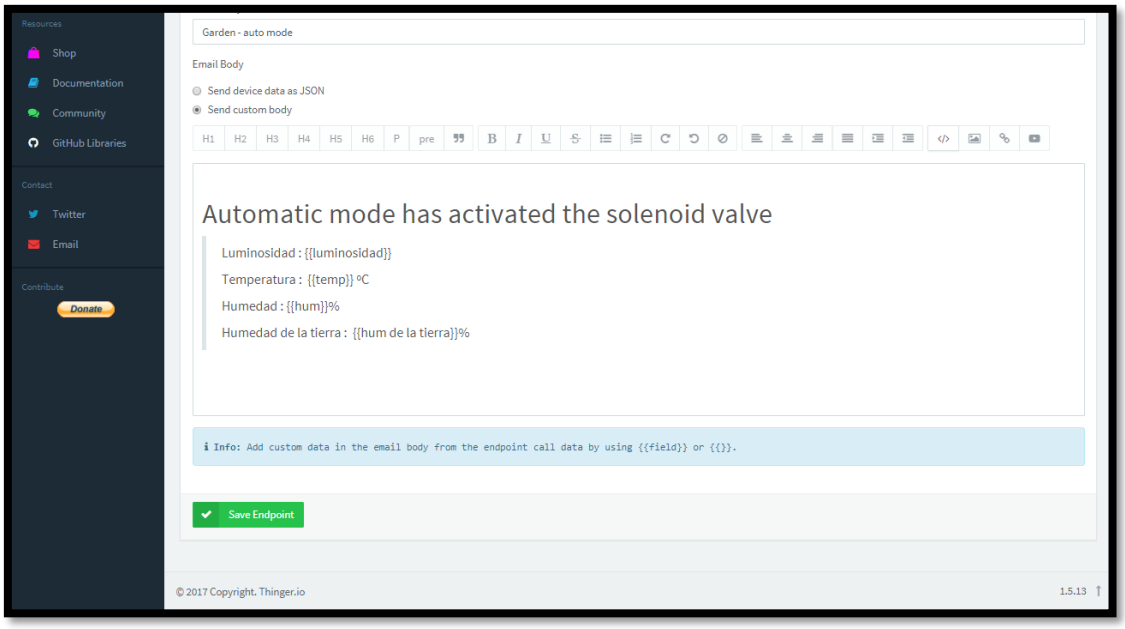


Figura 67. Endpoints 5 Thinger

A continuación, observamos el correo electrónico recibido desde Gmail, que se envió de manera automática por el sistema con la información correspondiente, cuando se cumplieron las configuraciones de horario establecidas por el usuario. El formato del mail, como ya hemos comentado, es el diseñado en la plataforma Thinger.io.

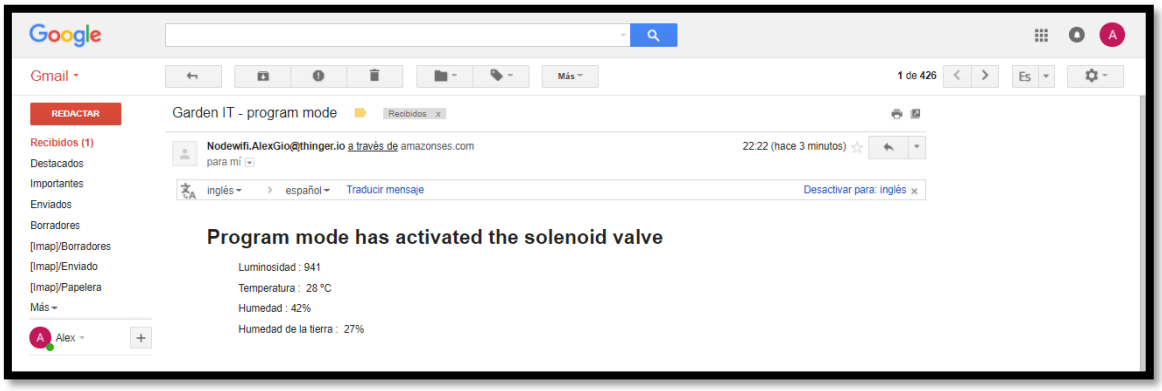


Figura 68. Email en Gmail.

5.8. Diseño y configuración XBee con XCTU

Aplicación para el diseño y configuración completa de los módulos XBee y las pruebas realizadas para verificar su correcto funcionamiento (21):

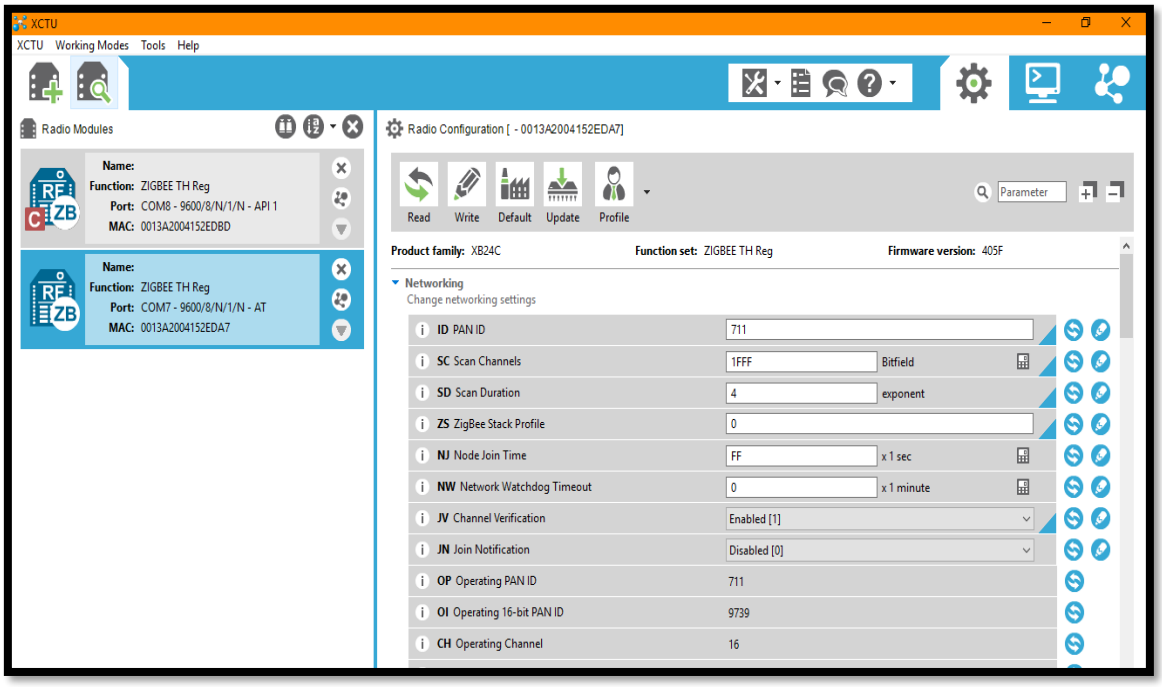


Figura 69. XCTU Configuración

En primer lugar, hay que tener en cuenta la arquitectura o diseño de la red inalámbrica que se desea establecer entre los módulos XBee, en este caso será en forma de estrella, es decir, un módulo central (coordinador) acoplado a la placa del microcontrolador que se comunica con el resto de módulos instalados en la tierra.

En el coordinador, lo más importante es establecer el PAN ID por el que vamos a identificar la red del dispositivo. A continuación, se habilita el modo coordinador y el modo API (Application Programming Interface).

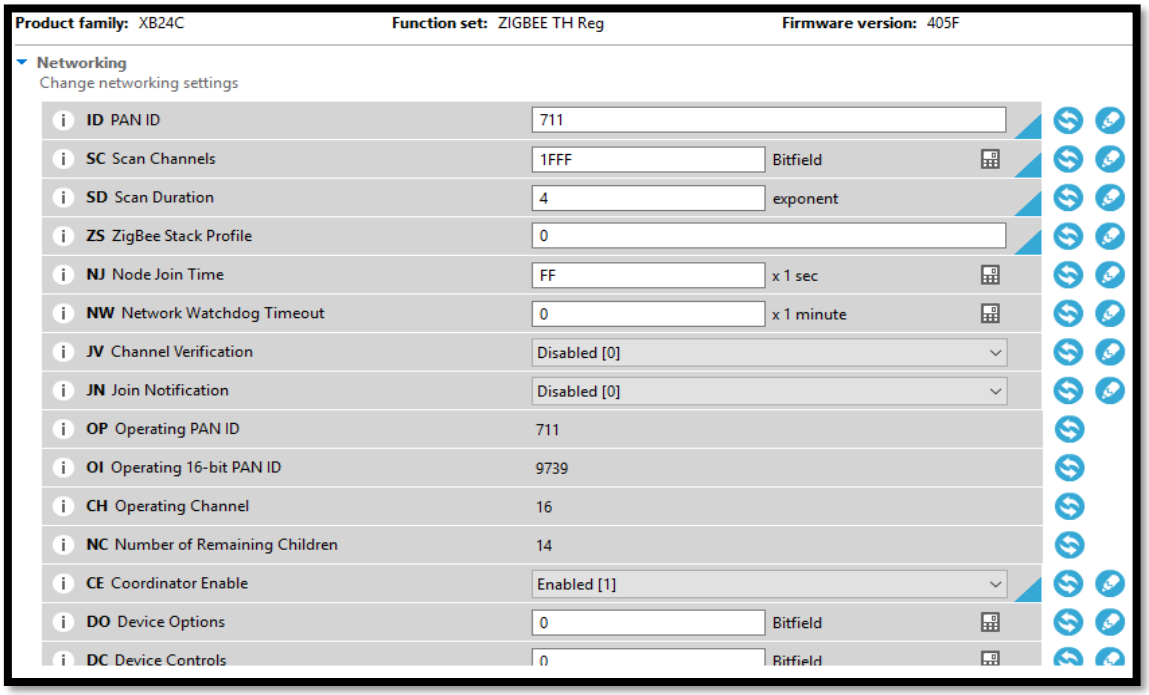


Figura 70. XCTU Configuración 2

También se pueden configurar el rango de la red, la seguridad, la frecuencia de transmisión, las entradas y salidas (analógicas, digitales), y muchas otras menos relevantes.

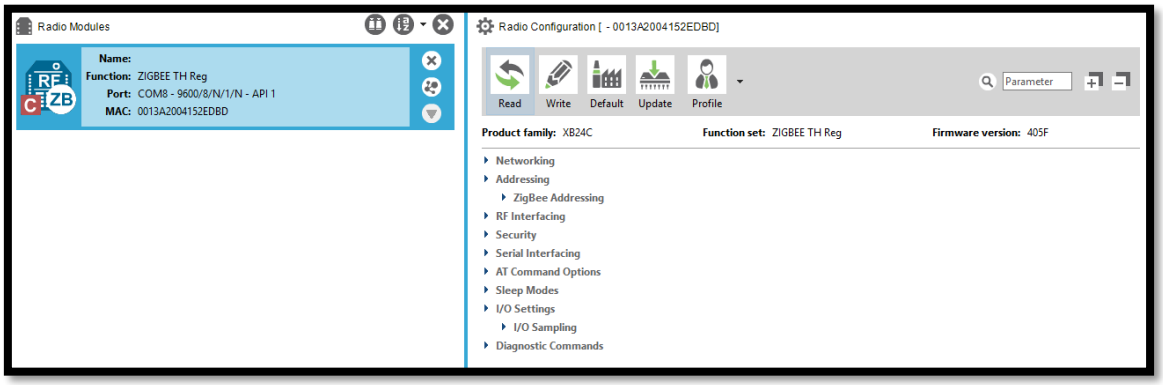


Figura 71. XCTU Configuración 3

En el módulo que actúa como extremo (“end point”), colocamos la misma PAN ID que en el coordinador y habilitamos la verificación de canales. Es necesario cambiar la dirección de destino porque en nuestro caso sólo comunicamos con el coordinador.

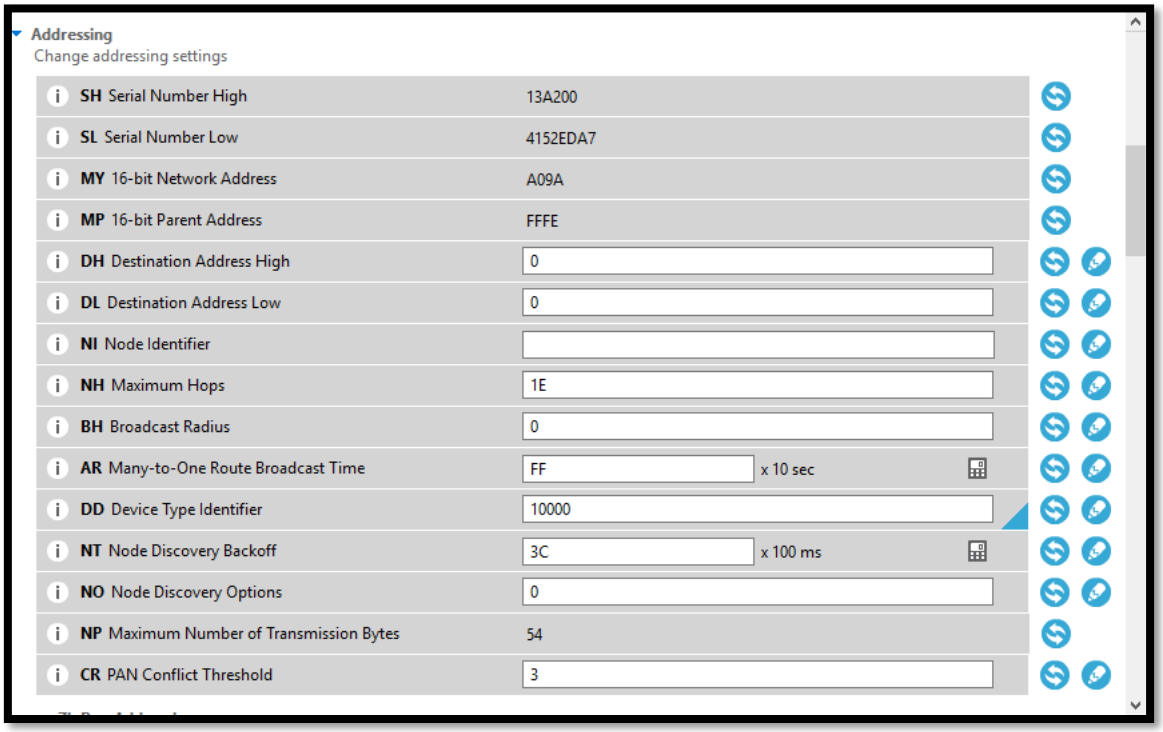


Figura 72. XCTU Configuración 4

En las entradas y salidas digitales, seleccionamos la entrada de AD0 y la colocamos como “ADC”, porque vamos a leer un sensor análogo (sensor de humedad de la tierra, YL-69).

Se puede establecer un periodo de tiempo para mandar la información o establecer que el módulo entre en modo suspensión (sleep), y que se despierte al pasar un tiempo determinado o se le soliciten los datos.

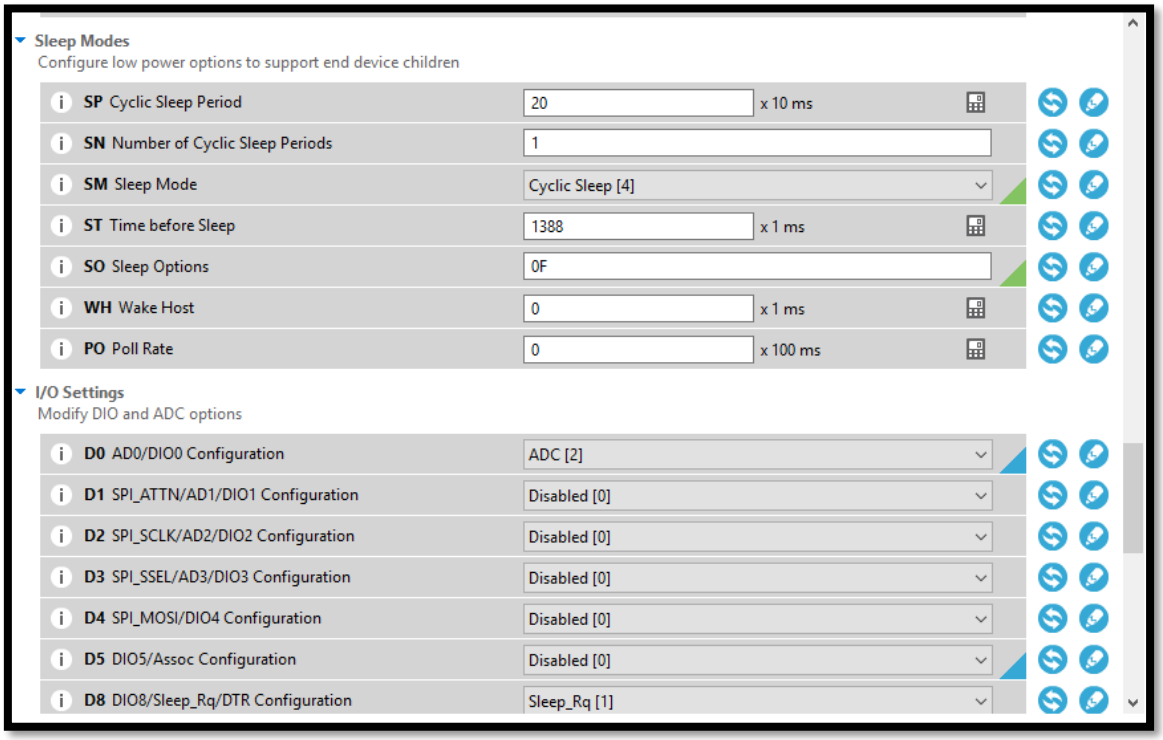


Figura 73.XCTU Configuración 5

Aquí observamos la comunicación entre los módulos XBee y la información que transmite al coordinador para posteriormente ser procesado por el microcontrolador y subido a la plataforma Thinger.

Mediante la configuración del módulo, podemos establecer el tipo de dato (analógico), el pin de entrada de los valores medidos por el sensor y la frecuencia de envío de mensajes.

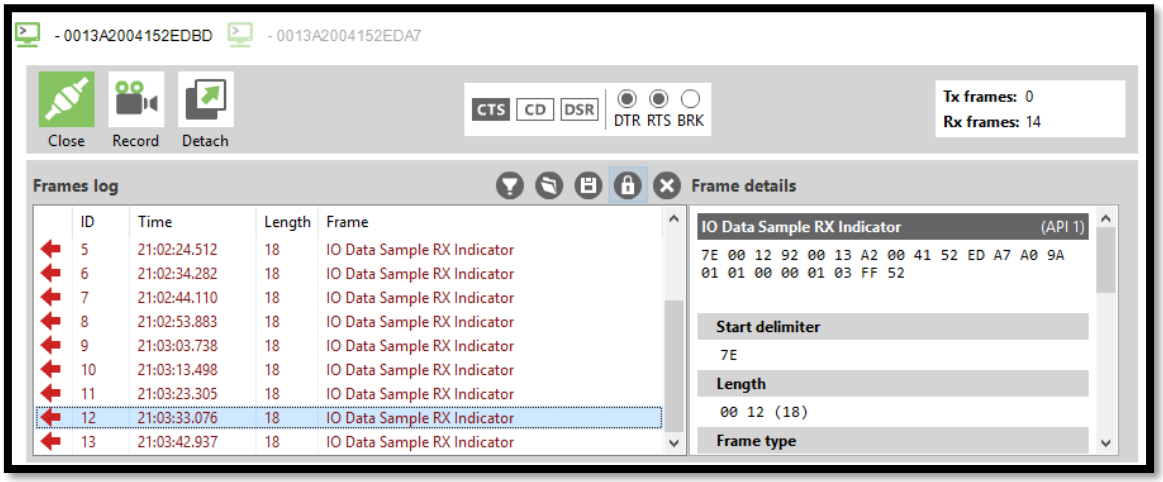


Figura 74. XCTU Configuración 6

Valor analógico recibido en el coordinador enviado por el dispositivo final que tiene integrado el sensor de tierra. Esta medida está hecha introduciendo el sensor en una tierra muy mojada, sirve como prueba y para calibrar el sensor y los umbrales.

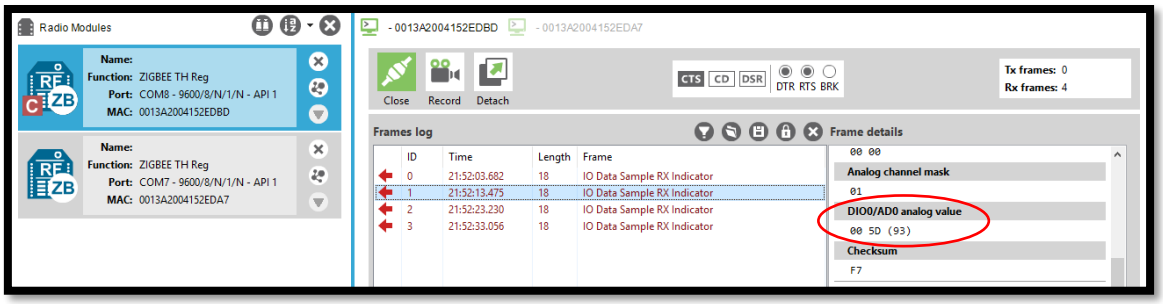


Figura 75. XCTU Configuración 7

6. Conclusiones y futuras líneas de investigación

A continuación, se detallan las conclusiones sacadas del proyecto, así como las dificultades encontradas a lo largo del proceso y las futuras líneas de investigación.

El objetivo que perseguía cuando me propuse este proyecto, ha sido superado con creces, no solo se ha creado un sistema de riego capaz de ser controlado a través de internet y de esta manera ayudar a mejorar la calidad de vida de las personas, reduciendo considerablemente el tiempo de implicación necesario para el mantenimiento del jardín y el uso del recurso del agua. También se ha dotado al sistema de inteligencia y funcionalidades que otorgan gran valor adicional, gracias al desarrollo en la plataforma Thinger.io, como las visualizaciones de los datos almacenados tanto en gráficas como los valores cuantitativos, el control automático del riego a partir de los parámetros recibidos, la programación del riego en un horario determinado y el envío de alertas al correo electrónico.

Pensar en una instalación para nuestro jardín mediante un sistema de riego automático ya no es una utopía. Significa una inversión inicial importante, que es una de nuestras desventajas o dificultades más relevante, pero a partir de ese momento supone una gran cantidad de ahorro de agua y la optimización del recurso hídrico. Pero, además, ahorrarás tiempo, trabajo y dinero en el futuro.

Es importante destacar que, a día de hoy, las personas que cuentan con jardín en sus hogares no tienen tiempo de cuidarlo, de regarlo cada día o de estar pendiente de si la humedad de la tierra es la adecuada para suministrar más agua o menos…

En mi opinión, estas ahorrando energía personal ya que, de este modo, ahorras preocupaciones y tiempo de estar pendiente de apagar o encender el riego y, sobre todo, si te vas de viaje esta preocupación desaparecerá.

Debemos destacar que precisamente ahora, en tiempos de temperaturas más cálidas se recomienda regar cuando el suelo no tenga la humedad y temperaturas adecuadas y deberás estar atento a estos cambios… pero con un sistema programado puedes olvidarte del problema.

Mediante el programador, podrás controlar cuando y durante cuánto tiempo se efectúa el riego. Junto al programador, emplearemos las válvulas, que son las que hacen que se produzca la apertura y cierre para que pueda pasar el agua.

Integrados a estos dos elementos está todo el equipo expuesto a lo largo del proyecto, estableciendo un coste a nivel Hardware de 146,84 € y el ordenador 72,92 €.

Podemos concluir con que los gastos no son muy elevados, pero debes invertir para poder tener la instalación adecuada, pero los resultados son asombrosos y el sistema aparentemente sencillo aporta gran valor en la calidad de vida de las personas alrededor del control y administración de su jardín.

Hay que destacar que el producto desarrollado es un sistema añadido que pretende facilitar la vida al usuario. En caso de existir un fallo en la comunicación del sistema a través de internet, el usuario siempre podrá activar su riego de forma física, quitando el obstaculizador de caudal de la electroválvula.

Uno de los **problemas** que he encontrado ha sido la cantidad de materiales necesarios para el desarrollo del proyecto y la realización de las pruebas, así como la integración de la electroválvula en un circuito de bajo voltaje, teniendo que comprar transformadores y baterías adecuados para cada elemento.

Otro de los problemas que he encontrado ha sido la necesidad de una estructura de jardinería para que funcione correctamente, presión del agua, tuberías de PVC, mangueras y demás materiales que se necesitan comprar para poder integrar todo en un sistema real.

Otro problema encontrado, ha sido la actualización del estado de la electroválvula en la plataforma. Esta información, se actualiza únicamente al cambiar el estado de abierto a cerrado o viceversa, pero existen problemas en la actualización del estado en el Data Bucket cuando se cambia repetidamente en un corto periodo de tiempo. Se entiende que este comportamiento no es el adecuado en la utilización del sistema y no es un problema relevante que afecte al correcto funcionamiento del sistema.

Como líneas futuras de investigación creo interesante el poder establecer este tipo de sistemas de riego en áreas más extensas, como por ejemplo un parques públicos o jardines de urbanizaciones, es fácilmente escalable integrando más módulos Xbee a la red del coordinador.

También creo interesante la instalación de este sistema de regadío en invernaderos, granjas, viñas o huertos de particulares, donde el nivel de humedad es elevado y con este tipo de sistemas podremos controlar todos los valores y regar desde nuestro hogar o incluso cualquier lugar con nuestro teléfono móvil con conexión a internet, podremos ayudar a invertir tiempo en otras labores que no sean los riegos y poder ahorrar dinero.

Podría ser interesante transformar la composición del módulo aislado formado por Zigbee y el sensor de la humedad de la tierra, para acoplar una placa fotovoltaica a la pila que lo alimenta

y así aumentar la duración de la batería dependiendo del sol. Creando un módulo más independiente.

Por último, creo que podría funcionar el integrar este sistema con una aplicación personalizada en Eclipse diseña a gusto del consumidor (“on demand”) vinculada a Thinger y sus funcionalidades.

7. Abstract

SUMMARY

Design, development and implementation of an intelligent irrigation system using a microcontroller, environmental sensors and Wi-Fi communication, which allows the control, manually or automatically, of the electro valve integrated in the system through the internet. Design of a graphical interface in the Cloud platform (Thinger.io), where the user can control, monitor the data collected by the sensors, and configure the functions of the irrigation system.

INTRODUCTION

Context

Nowadays, work and daily life cover much of our time and with the new technologies, mobile devices and communications many of the tasks have been optimized and done more efficiently.

There are numerous private and public gardens that require care and maintenance. This project aims to install a hardware component that monitors and communicates the state of a given garden from a circuit of controllers and sensors and a Software component, by which we can control these parameters and execute orders to make valve openings or closures, to irrigate our garden if necessary.

In this application the user will have access in a simple and efficient way to his domotic garden, having absolute control of the irrigation system and the collected data by the sensors, being able to configure the irrigation and other functionalities his/her own way.

Motivation

Today, technology is almost available to all, and allows us to create tools and services to solve problems and thus spend the time on other tasks.

The main reason for this project development is to improve life quality of users, and to manage the gardens in a comfortable and fast way, thus investing time in other tasks. In addition, it highlights the great importance of mobile technology and ease of use, which makes many of us spend every day some time near any of these devices, whether for personal issues, work, communication or other services.

goals

The objective of this project is to create a Hardware component to be installed in a simple way in the gardens, to monitor the parameters of the place and allowing user to manage and control in a simple and effective way the state of his/her garden from the application with the corresponding Software.

For this purpose, two parts are used:

- Hardware:

• Create a functional hardware composed of temperature, humidity and other sensors and controllers.

• Communication in the controllers to transfer the information collected by the sensors and receive the information from outside.

• Intelligence in the controllers to be able to interpret and execute the opening and closing orders of the electromagnetic valves in charge of the passage of water.

- Software:

• Manage information: The user can check the status of the place receiving the information by temperature and humidity sensors.

• Monitoring and recording of data: The user can check the history of the data and see the evolution graphically.

• Execute orders: The user can execute orders to open or close valves in the garden to irrigate the place.

• Program irrigation: The user can program the irrigation at a specific time and date.

• Irrigation automation: The user can establish an automatic mode of irrigation under certain critical conditions.

STATE OF THE ART

The objective of this section is to establish a list of advantages and disadvantages of the different technologies that will be used during the project, in order to establish which are optimal for our objective. In addition, a market study will be carried out to verify the feasibility of the project.

Starting point of the project

When we talk about gardening, the necessary care and maintenance routinely makes it tedious, having to make trips to the indicated areas and proceed to irrigate the place. With the evolution of technology and mobile devices, many of these tasks and others of such characteristics have been optimized (1).

More and more people are using the technology and mobile devices, and they take advantage of the services they offer (1).

In 2016 a report on the connectivity and access to information society was made, which reflects the rapid digitalization of our society.

The 78.7% of the Spanish population regularly connects to the internet, being the mobile phone the main device through which they connect (88.3%) while computers and laptops are in second place (78.2%) (1).

Choice of technologies

The idea is to have a desktop application with which the user can access all the services offered and interact with them, although the possibility of creating a mobile application for Android later on, will be considered.

A good choice would be Eclipse, which offers great possibilities in our development and tools like WindowBuilder (22), which we are able to develop a GUI (Graphic User Interface) easily and quickly with, thanks to the facilities it offers in the development of this type of applications, adding controls with their drag-and-drop functions, adding events generating the code automatically. In addition, projects with these tools have already been carried out previously, reason why the learning cost is practically null.

Another option would be a web server where install an application with html5 visualization, with CSS3 and bootstrap code to adapt the interface to the different devices.

However, having studied the possibilities offered by Thinger, its functionalities and services, it would be illogical to ignore this existing tool that offers us so many facilities. Therefore, we will

omit in the first instance the section of application development or graphical user interface, using for it, the web platform Thinger (20).

For the development of the NodeMCU controller, we will use the development environment facilitated by the Arduino brand (Arduino Software IDE) (19).

GENERAL DESCRIPTION

Product perspective

After carrying out the market study of the different companies and services that offer and analyze the most relevant components for the development of our system, we can begin to develop our project with the conclusions obtained in an optimal way and in accordance with our needs and technological and economic limitations.

The irrigation system consists in controlling the environmental parameters of temperature, humidity and luminosity and, depending on the levels, be able to activate or deactivate the solenoid valves to proceed to irrigate the area.

The system will have three modes of operation:

• Manual operation: Allows the user to remotely control the activation or deactivation of solenoid valves.

• Automatic operation: The system acts accordingly to the soil moisture levels received, if they are below a defined threshold and the climatic conditions are adequate (low temperature, no precipitation, low luminosity, and day time appropriate), will activate the solenoid valves automatically.

• Programmed operation: The system acts accordingly to the time set by the user and will activate the solenoid valves automatically the irrigation time established.

NodeMCU (23), Zigbee (10) controllers, sensors, solenoid valves and other components will simulate the operation of a real irrigation system. An interface will be configured using Thinger, which its operation will be controlled through.

Software Scope

The system will focus on two main elements:

Monitoring system: Monitoring and control software through Thinger (or develop an interface) with the necessary functionalities to control the irrigation system.

The main objective is focused on facilitating the management of irrigation by the sistem, controlling the environmental parameters and giving more information about the state of the place and being able to decide with more knowledge about environment, the actions that have to be carried out.

Automatic system: Automatic control software programmed in the NodeMCU board (23). This software will be developed in the Arduino programming environment, this system will automatically control the solenoid valves according to the parameters established and measured by the board and other sensors transmitted by the Xbee modules (10).

General Capabilities

The main capabilities that the system will have are:

• The system allows the user to activate or deactivate the automatic system and define its critical thresholds from which the system must act automatically.

• The interface will have a panel in which the user will be informed of the status of the solenoid valve and the environmental parameters measured.

• The automatic system has as a rule to yield to any manual action performed by the user, aborting their automatic performance.

• At the time communication between the system and actuators or sensors fails, the automatic system will be rendered useless without generating any unsafe behaviour.

General Restrictions

The system must be tolerant to failures that must be controlled, as any failure of the system would suppose a serious danger of flood or drought that, in any case, would compromise the state of the place´s flora.

The correct use of the irrigation system is delegated completely to the user, who will have complete freedom and will be responsible for its correct use.

The automatic system must be configured with threshold values less than or equal to 25% ground moisture (15% by default).

The activation of solenoid valves in automatic mode will not be allowed if the conditions are not optimal for irrigation. High unfavourable temperatures and brightness (temp:> = 20 ° C, lum:> = 500) or a time from 11:00 to 18:00 are considered unfavourable conditions.

Activation of solenoid valves in program mode is not permitted if the soil moisture is higher than 60% or it is raining at that time.

The system will never be able to act in automatic or programmed mode without a user who approves or configures these actions (by default deactivated).

Operating Environment

The execution environment of the system configuration and control application will be a computer with a Windows 10 operating system or higher and a jdk-8u60 java environment or compatible in Eclipse and internet connection.

On the other hand, the code that will transform the different signals of the actuators into actions will be carried out on a physical NodeMCU ESP8266 1.0 Amica (23) and ZigBee (10) modules.

ESTIMATED RESOURCES

Section oriented to detail the estimation of software, hardware and human resources needed to complete the project.

Hardware Resources

The project is designed for an Arduino architecture or compatible microcontroller, with sensors that collect information. For its correct operation, we need numerous hardware resources of all types:

• A computer, not necessarily high-performance, but one that allows you to work seamlessly with code development and testing environments and with USB and wireless connections.

• Arduino or microcontroller, preferably with internet connectivity, by ethernet or wifi cable, microUSB connection to connect to a computer for simple programming and sufficient input / output pins for external connections to sensors and other elements of the circuit.

• Sensors that take measurements related to climate and land, such as temperature, humidity, lightness and rainfall.

• Cables, resistors and other materials (relays, diodes, transistors, transformers ...) for the development of a correct and stable electrical system.

• Router or connection to the network to communicate over the internet and to upload data and interact with the system.

• ZigBee modules (10) for sending information wirelessly.

Software Resources

For the development of the project at Software level will use Open Source or free tools and applications that, although requiring some kind of license, are accessible to members of University for free through agreements with Microsoft Imagine (24).

It is necessary to part of the basic software of the computer, which is our main tool for the development of the project, as well as a preferably updated operating system (24):

• Tool for the development of documentation for the memory and user manual.

• Development environment and tests for microcontroller / Arduino programming.

• Development and testing environment for programming the XBee modules.

• Tool for the elaboration of a presentation for the exhibition of the project.

Human Resources

The project is carried out by a single person who is responsible for all the tasks to be carried out from its beginning to its delivery, and subsequent maintenance if necessary until its withdrawal. It will have the help of a tutor assigned to the project that will act as "Product owner" or client, to guide the student in the elaboration of the project.

ARCHITECTURE

For the development of this project, it has been considered to apply a software architecture pattern based on: Model - View - Controller (MVC). This architecture is based on code reuse and separation of concepts, positively influencing the ease of development and maintenance.

- Model: It is the layer where you work with the data, will contain mechanisms to access the information and to update its status. The data will usually be in a database. In our case, it consists of the data stored in the Data Buckets, which are sent by the controller.

- View: Contains the code of our application that will produce the visualization of the user interfaces. The user interacts with the different views to request information or execute actions in the irrigation system.

- Controller: Contains the necessary code to respond to the actions that are requested in the application, serves as a link between the views and the models, responding to the mechanisms that may be required to implement the needs of the system. The controller is in charge of sending the information collected by the sensors, and the physical activation of the solenoid valve.

DEVELOPMENT VIEW

The Arduino component will carry out the execution of the irrigation system in either manual, automatic or programmed mode. It is the software integrated in the NodeMCU board (23) and will handle the communication with Thinger platform, collect the values of the sensors and act on the solenoid valves when necessary or indicated from the SW (Web interface / Thinger) (20) hosted on the Internet (Web Server).

The Control component that will be responsible for telling Arduino the active mode of operation, as well as performing the manual irrigation control (enable / disable electro valve).

The monitoring component that will be responsible for displaying the data you receive from the Arduino component in the graphical interface (data buckets and dashboards).

The Configuration component will allow you to configure the thresholds of the sensors for irrigation in automatic mode, the irrigation time and the hour and minute of the programming mode.

The Alerts component will be responsible for notifying the user via email when the solenoid valve is automatically activated by automatic mode or by programming.

CIRCUIT DESIGN (FISICAL VIEW)

Central module (coordinator)

To get the ambient luminosity, a single voltage divider is created, a resistance of 1KΩ is used and a LDR connected to the 3V output of the microcontroller, the current flows through the LDR which its resistance will oscillate depending on light upon it. The LDR output Is connected to an analogical pin of the microcontroller (A0) to evaluate the sensor measurement.

To check the state of precipitation, the YL-38 module connected the altimeter YL-83 is used. The YL-38 sensor transmits the difference of the potential measurement through its conductive tracks. The YL-38 control module integrated by a pre-calibrated potentiometer which compares the voltage and sends a digital output signal (0/1) according to the pin of the microcontroller (D3).

The DHT-22 is used for the temperature and humidity readings, it is integrated in a module with resistors and a LED of ignition, reason why its connections are very simple. Sensor´s data output is connected to the digital pin (D7) of the board, a special library is required to read these digital data.

For the control of the solenoid valve a relay is used, integrated in a module with resistors and other current limiters like diode and transistor to avoid that the current flows back and burns the circuit. It is connected to the board, and through the pin (D5) of the board the switch activation is controlled, which closes the circuit of its other end and activates the solenoid valve, located at the "normally closed" output, powered by a 24V transformer.

Serial communication is use between the microcontroller and the module XBee coordinator. The XBee modules operate at 3.3V and the pins are not 5V tolerant. From the microcontroller can power an XBee module, but the serial communication is a 5V and in the module XBee is a 3.3V. Therefore, we use a voltage divider.

Final module (final device)

To check the soil moisture, we use the YL-38 module connected to the YL-69 sensor. The YL-69 sensor transmits the potential difference measured by its conductive tips, the YL-38 controller module compares the voltage and sends an analogical output signal (0-1023) to the pin (A0) of the XBee module.

ZIGBEE´S DESIGN AND ARCHITECTURE

An XBee network is basically formed by 3 types of elements. A single Coordinator device, Routers and End devices. The XBee modules are versatile for establishing different network topologies, depending on the XBee series that we choose, different networks can be created (31).

The Coordinator: It is the node of the network that has the unique function of forming a network. It is responsible for establishing the communications channel and the PAN ID (network identifier) for the entire network. Once these parameters are established, the Coordinator can form a network, allowing devices to join Routers and End Points. Once the network is formed, the Coordinator acts as Router, that is, participate in packet routing and be the source and / or receiver of information.

The Routers: It is a node that creates and maintains information about the network to determine the best way to route an information packet. Logically, a router must join a Zigbee network before it can act as a Router by retransmitting packets from other routers or Endpoints.

End Device: End devices are unable to route packets. They must always interact through their parent node, be it a Coordinator or a Router, that means, it can not send information directly to another end device. Usually these devices are fed by batteries. The consumption is less when not having to perform routing functions and being able to sleep.

For our system, a star-like architecture has been implemented, cause it is a simple and scalable integration where only the central module is connected to the microcontroller, which is the coordinating XBee module, and one or more final devices to gather information of certain lands or gardens. This way, we can perform the monitoring of different garden extensions from the same microcontroller.

CONCLUSIONS AND FUTURE LINES OF RESEARCH

The conclusions obtained from the project, as well as the difficulties found throughout the process and the future lines of research, are detailed below.

The objective that I pursued when I proposed this project, has been exceeded, not only an irrigation system capable of being controlled through the internet has been created. Life quality of users has been improved, reducing considerably the time of human involvement necessary for the maintenance of the garden and the use of water resources. Intelligence and lots of functionalities has also been endowed to the system with great added value, thanks to the development in the Thinger.io platform, such as visualizations of the data stored (graphs and quantitative values), automatic irrigation control from the parameters received, scheduling the irrigation at a certain time and sending alerts to the email.

Thinking about an installation for our garden by means of an automatic irrigation system is no longer a utopia. It implicates an important initial investment, which is one of our most relevant disadvantages or difficulties, but from that moment on, it entails a great amount of water saving and optimization of water resource. But, in addition, you will save time, work and money in the future.

It is important to highlight that, now a days, people who have garden in their homes do not have time to take care of it, to water it every day or to be aware of ground humidity is adequate to supply more or less water ...

In my opinion, you are saving personal energy since, in this way, you save worries and time to be aware of turning off or on the watering and, above all, this system will be extremely useful if you go out for holydays, cause all this worries will disappear.

We must emphasize that now, in times of warmer temperatures it is recommended to water when the soil does not have adequate humidity and temperatures and you must be aware of these changes ... but with a programmed system you can forget about the problem.

Using the programmer, you can control when and for how long the irrigation takes place. Next to the programmer, we will use the valves, which are the ones that cause the opening and closing to occur so that the water can pass through.

Integrated to these two elements is all the equipment exposed throughout the project, establishing a hardware level cost of 146.84€ and the computer 72.92€.

We can conclude that the expenses are not very high, but you must invest to be able to have the proper installation, but the results are amazing and the seemingly simple system brings great value in the people´s life quality around the control and management of your garden.

One of the problems I have found has been the amount of materials needed for the development of the project and the realization of the tests, as well as the integration of the solenoid valve in a low voltage circuit, having to buy transformers and batteries suitable for each element.

Another problem I have encountered has been the need for a gardening structure to work properly, related with PVC pipes and hoses and other materials needed to be purchased, in order to integrate everything into a real system.

As future lines of research, I think it is interesting to be able to establish this type of irrigation system in larger areas, such as public parks or urbanization gardens, it is easily scalable by integrating more Xbee modules to the coordinator network.

I also find interesting the installation of this irrigation system in greenhouses, farms, vineyards or private orchards, where weather parameters are critical and with this type of systems, we can control all values and water from our home or even any place with our smartphone with internet connection, we can help to invest time in other tasks and save money.

Could be interesting, to transform the End device module composition, form by with Zigbee and the ground moisture sensor, to attach a photovoltaic board to the battery which feeds it and thus increase the battery life depending on the sun. Creating a more independent module.

Finally, I think it could work to integrate this system with a custom application in Eclipse designs on client demands, linked to Thinger and its features.

8. Bibliografía

1. El móvil supera por primera vez al ordenador para acceder a internet. El Mundo. 2016 Abril.
2. Ranch Systems. [Online]. [cited 2017 junio. Available from: http:///www.ranchsystems.com/home/.
3. Samcla. [Online]. [cited 2017 junio. Available from: http://www.samcla.com/es/empresa.html.
4. Libelium. [Online]. [cited 2017 junio. Available from: http://www.libelium.com/.
5. Arduino.CC. [Online]. [cited 2017 mayo. Available from: www.Arduino.cc.
6. NodeMcu. [Online]. [cited 2017 mayo. Available from: http://nodemcu.com/index_en.html.
7. González AG. DHT22: Sensor de humedad/temperatura de precisión para Arduino. PanamaHitek. 2014 agosto; 8.
8. González AG. Módulo HL-69: Un sensor de humedad de suelo. PanamaHitek. 2014 abril; 9.
9. González AG. Módulo YL-83: Un detector de lluvia. PanamaHitek. 2014 abril; 6.
10. EL MÓDULO BLUETOOTH HC-05. [Online]. [cited 2017 julio. Available from: www.prometec.net.
11. Zigbee Alliance. [Online]. [cited 2017 julio. Available from: http://www.zigbee.org/.
12. Young JK. A Practical Guide to Battery Technologies for Wireless Sensor Networking. Sensors. 2008 julio; 1.
13. Rain Bird. [Online]. [cited 2017 mayo. Available from: https://www.rainbird.es/productos/kit-de-control-filtros-valvulas-y-reguladores-de-presion/electrovalvula-de-caudal-bajo.
14. Prometec. [Online]. [cited 2017 mayo. Available from: http://www.prometec.net/esp8266/.

15 Arduino. [Online]. [cited 2017. Available from: https://store.arduino.cc/arduino-wifi-shield.
.

16 Android Studio. [Online]. [cited 2017 mayo. Available from:
. https://developer.android.com/studio/index.html?hl=es-419.

17 Xcode Developer. [Online]. Available from: https://developer.apple.com/xcode/.
.

18 Microsoft Visual Studio. [Online]. Available from:
. https://www.visualstudio.com/es/?rr=https%3A%2F%2Fwww.google.es%2F.

19 Eclipse IDE. [Online]. [cited 2017. Available from: https://eclipse.org/ide/.
.

20 Arduino. [Online]. Available from: https://www.arduino.cc/en/main/software.
.

21 Thinger.io. [Online]. Available from: https://thinger.io/.
.

22 XCTU. [Online]. Available from: https://www.digi.com/products/xbee-rf-solutions/xctu-
. software/xctu.

23 Eclipse WindowBuilder. [Online]. Available from: https://eclipse.org/windowbuilder/.
.

24 NodeMcu. [Online]. Available from: http://nodemcu.com/index_en.html.
.

25 Microsoft Imagine. [Online]. Available from:
. https://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?cmi_cs=1&cmi_m
nuMain=bdba23cf-e05e-e011-971f-0030487d8897&ws=8738733a-6d9b-e011-969d-
0030487d8897&vsro=8.

26 Polaridad. [Online]. Available from: https://polaridad.es/ldr-fotorresistencia-luz-
. luminosidad-medir-medicion-arduino/.

27 Windows 10. [Online]. Available from: https://www.microsoft.com/es-es/windows/features.

28 Microsoft Word 2016. [Online]. Available from: https://products.office.com/es-es/word.

29 Microsoft PowerPoint 2016. [Online]. Available from: https://products.office.com/es-es/powerpoint.

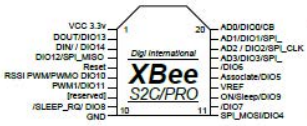
30 Microsoft Visio. [Online]. Available from: https://products.office.com/es-es/visio/flowchart-software?tab=tabs-1.

31 Microsoft Projet Porfesional. [Online]. Available from: https://products.office.com/en-us/project/project-professional-desktop-software.

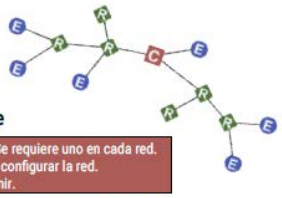
32 Digi.com. [Online]. [cited 2017 junio. Available from: https://www.digi.com/resources/documentation/digidocs/PDFs/90000976.pdf.

9. Anexos

9.1. Anexo 1: Poster XBee



XBee es un microcontrolador fabricado por digi, el cual utiliza el protocolo Zigbee. XBee utiliza 3.3V y tienen un espaciado entre pines menor que el utilizado en un protoboard. Dado lo anterior es recomendable utilizar una tarjeta o kit para que sea más fácil su uso.



Roles XBee

Coordinador: Se requiere uno en cada red. Se encarga de configurar la red. No puede dormir.

Router: Pueden existir multiples en una red. Pueden redirigir los mensajes a otros routers o End Devices. No pueden dormir.

End Device: Pueden existir muchos, no pueden redirigir mensajes. Pueden dormir para ahorrar energía.

	Especificaciones		
S2C	Voltaje de operación: 2.1 – 3.6V Corriente de operación: 33mA@3.3V Rango en interiores: 60 metros Rango en línea vista: 1200 metros Max valor lectura analógica: 1.2V	Pines I/O: 13 Entradas analógas: 4 *Configurable como red mesh* *Red auto reparable*	Firmware: Zigbee, Digimesh, 802.15.4 Velocidad de transmisión RF: 250kbps Frecuencia: 2.4GHz - 2.5GHz Temperatura operación: -40 a 85°C
S2C PRO	Voltaje de operación: 2.7 – 3.6V Corriente de operación: 31mA@3.3V Rango en interiores: 90 metros Rango en línea vista: 3200 metros Max valor lectura analógica: 1.2V	Pines I/O: 13 Entradas analógas: 4 *Configurable como red mesh* *Red auto reparable*	Firmware: Zigbee, Digimesh, 802.15.4 Velocidad de transmisión RF: 250kbps Frecuencia: 2.4GHz - 2.5GHz Temperatura operación: -40 a 85°C

Modos XBee

Transparente: Los dispositivos actúan como un reemplazo de cable serial. Cuando los datos RF son recibido, el dispositivo envía los datos a través del puerto serie. Utilice la interfaz de modo de comando AT para configurar los parámetros del dispositivo.
Comando: Basada en tramas, amplía el nivel en que una aplicación host puede interactuar con las capacidades de red del dispositivo. Cuando esta en modo API, el dispositivo contiene todos los datos que entran y salen en marcos que definen operaciones o eventos dentro del dispositivo.

Setup XBee

Conecta el Xbee a un adaptador TTL a Serial como un FTDI
Utiliza el software gratuito X-CTU para configurar el módulo XBee
Baud:9600 – FC: Hardware – Data Bits 8 – Parity: None – Stop Bits: 1

Ajustes Básicos

PAN ID: es la red a la cual se conectará el módulo. Si es 0, el XBee se asociará a cualquiera que esté disponible
DH/DL: Es la dirección del módulo de destino. Se utiliza para enviar información a un XBee en específico. Si se configura en 0 enviará datos solo al coordinador. Si se configura en 0x000000000FFFF hará un broadcast (envío a todos los módulos de la red)

Ajustes Pin

Para poder trabajar con los pines como entradas/salidas en un XBee, debe estar configurado en modo API.
D0 - Configura el pin en 0 para comenzar a leer datos
IR - realiza una lectura del pin cada XX milisegundos

Trama API para envío de comandos AT remotos

Byte	Ejemplo	Descripción
0	0x7E	Byte de inicio – indica el comienzo del paquete de datos (frame)
1	0x00	Largo – Número de bytes (ChecksumByte# – 1 – 2)
2	0x10	
3	0x17	Tipo de mensaje - 0x17 significa que es solicitud de comando AT
4	0x01	Frame ID – secuencia del paquete
5	0x00	Dirección de destino de 64-bi t(número de serie)
6	0x13	MSB es el byte 5, LSB es el byte 12
7	0xA2	
8	0x00	0x0000000000000000 = Coordinador
9	0x40	0x000000000000FFFF = Broadcast
10	0x8B	
11	0x78	
12	0x4E	
13	0xFF	Dirección de la red de destino
14	0xFE	(configúralo como 0xFFFE para enviar un bodcast)
15	0x02	Opción del comando remoto (configuralo como 0x02 para aplicar los cambios)
16	0x44 (D)	Nombre del comando AT (Dos caracteres ASCII)
17	0x34 (2)	
18	0x05	Parámetro del comando
19	0x25	Checksum

Conexión con Arduino:
Arduino TX se conecta la RX de XBee (Data IN)
Arduino RX se conecta al TX de XBee (Data Out)

Integración con Arduino:
Los datos enviados utilizando Serial.print() saldrán por el puerto TX del Arduino, que estará conectado al RX del módulo XBee. Si XBee está en modo AT, se transmitirá inalámbricamente hacia el destino. Los datos recibidos en el módulo XBee serán enviados al puerto serial.

Ejemplo para Arduino: Lectura de un valor análogo utilizando modo API

```
// XBee remoto: AT, XBee base: API
if (Serial.available() >= 21) { // Nos aseguramos que ha llegado el mensaje completo
if (Serial.read() == 0x7E) { // 7E es el byte de inicio
  for (int i = 1; i<19; i++) { // descartamos los bytes hasta llegar los datos análogos
    byte discardByte = Serial.read();
  }
int analogMSB = Serial.read(); // Lee el primer byte del dato análogo
analogLSB = Serial.read(); // Lee el segundo byte del dato análogo
int analogReading = analogLSB + (analogMSB * 256); }
}
}
```

Ejemplo Arduino: Cambiar la configuración de un pin en un XBee remoto

```
// XBee remoto: AT, XBee base: API
Serial.write(0x7E); // byte de inicio
Serial.write((byte)0x0); // Largo MSB (siempre 0)
Serial.write(0x10); // Largo LSB
Serial.write(0x17); // 0x17 es el tipo de mensaje para enviar comandos AT
Serial.write((byte)0x0); // Frame ID (no solicitamos repuesta)
Serial.write((byte)00); // Envía los 64 bit de la dirección de destino
Serial.write((byte)00); // (Enviando 0x000000000000FFFF (broadcast))
Serial.write((byte)00);
Serial.write((byte)00);
Serial.write((byte)00);
Serial.write((byte)00);
Serial.write(0xFF);
Serial.write(0xFF);
Serial.write(0xFF); // Red de destino
Serial.write(0xFE); // (enviar 0xFFFE si es desconocida)
Serial.write(0x02); // configurar 0x02 para aplicar los cambios
Serial.write('D'); // Comando AT : D1
Serial.write('1');
Serial.write(0x05); // Configura D1 para ser 5  (Digital Out HIGH)
long chexsum = 0x17 + 0xFF + 0xFF +0xFF + 0xFE + 0x02 + 'D' + '1' + 0x05;
Serial.write( 0xFF – (chexsum & 0xFF)); // Checksum
```

Trama API para recepción de datos I/O

Byte	Ejemplo	Descripción
0	0x7E	Byte de inicio - indica el comienzo del paquete de datos (frame)
1	0x00	
2	0x14	
3	0x92	Tipo de frame 0x92 indica que es un muestreo de las entradas del XBee
4	0x00	Dirección de origen de 64-bit (número de serie)
5	0x13	MSB es el byte 4, LSB es el byte 11
6	0xA2	
7	0x00	
8	0x40	
9	0x8B	
10	0x78	
11	0x4E	
12	0xA4	Dirección de 16-bit de la red de origen
13	0x02	
14	0x01	Opciones de recepción: 01 = Packet acknowledged 02 = Broadcast packet
15	0x01	Número de muestras. Siempre debe ser 1 dadas las limitaciones de XBee
16	0x00	Máscara para el canal digital, indica que pines estan configurados como DIO
17	0x30	
18	0x01	Máscara para el canal análogo, indica cuales pines están configurados como ADC
19	0x00	Lectura de los canales digitales. Estos dos bytes contienen los estados de
20	0x20	los pines configurados como DIO
21	0x02	Lectura del canal análogo.
22	0x0C	Cada canal entrega 2 bytes con el resultado de la lectura del ADC
23	0x20	Checksum (0xFF - la suma de todos los bytes desde el byte 3 a hasta el último)

Modo Sleep

End Device puede dormir para ahorrar energía. Un End Device que solo despierta cada 5 minutos para enviar datos puede solo estar despierto por 6 segundos en un día.
SM – 4 = Cyclic Sleep
SP - Sleep time (hasta 28 segundos)
SN - Número de ciclos sleep
ST – Tiempo que permanecerá despierto

Pin I/O Opciones

0 –Disabled
1 –N/A
2 –ADC
3 –Digital IN
4 –Digital OUT, LOW
5 –Digital OUT, HIGH

Máscara para canal digital

Primer Byte
n/a n/a n/a D12 D11 D10 n/a n/a
Segundo Byte
D7 D6 D5 D4 D3 D2 D1 D0
Ejemplo:
0x00 0x13 = 0000 0000 0000 1101
Pins D3, D2 y D0

Máscara para canal análogo

(volt) n/a n/a n/a A3 A2 A1 A0
Ejemplo:
0x05 = 0000 0101 = Pin A2 and A0

9.2. Anexo 2: Código Arduino integrado en el microcontrolador

```
#include <time.h>
#define _DEBUG_
#include <SPI.h>
#include <ESP8266WiFi.h>
#include <ThingerWifi.h>
#include "DHT.h"
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <SoftwareSerial.h>



#define USERNAME "AlexGio"
#define DEVICE_ID "Nodewifi"
#define DEVICE_CREDENTIAL "emCeG******"
#define SSID "Orange-8335"  //wifi
#define SSID_PASSWORD "26E*****"  //wifi password

#define DHTTYPE DHT11   // DHT 11
#define DHTPin D7
#define rele D5
#define rain D3
#define LDR A0

DHT dht(DHTPin, DHTTYPE);
ThingerWifi thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);
SoftwareSerial XBee(3,2);
long time1;
bool aux = false;
bool previous = false;
bool modo = false;
```

```
bool program = false;
int watertime = 10;  //default irrigation time
int minute = 0;
int hour = 0;
bool auxClock = false;
int auxminute;
int umbraltierra = 15; // default: 15% - 1000 es muy seco
int tierra ;
int isRain = 1;

void setup() {
  Serial.begin(9600);
  XBee.begin(9600);
  thing.add_wifi(SSID, SSID_PASSWORD);
  pinMode(LDR,INPUT);
  pinMode(DHTPin,INPUT);
  pinMode(rain,INPUT);
  pinMode(rele,OUTPUT);
  digitalWrite(rele, LOW);

  thing["rele"] << [](pson& in){
    if(in.is_empty()){
        in = aux;
    }
    else{
      aux = in;
      digitalWrite(rele, in ? HIGH : LOW);
      if (aux==true){
          auxminute = timeClient.getMinutes() + watertime;
          if (auxminute > 59) auxminute = auxminute-60;
      }
    }
  };
```

```
thing["program"] << [](pson& in){
    if(in.is_empty()) in = program;
    else program = in;
};
thing["modo"] << [](pson& in){
    if(in.is_empty()) in = modo;
    else modo = in;
};
thing["clima"] >> [](pson& out){
    out["temp"] = dht.readTemperature();;
    out["hum"]=  dht.readHumidity();;
    out["luminosidad"] = analogRead(LDR);
    out["hum de tierra"]=  tierra;
};

thing["hour"] << [](pson& in){
  if ((int)in>0 && (int)in<=24 ){
    if((int)in==24) hour = 0;
    else hour = (int)in;
  }else{
    in = hour;
  }
};
thing["minute"] << [](pson& in){
  if ((int)in>0 && (int)in<=60 ){
    if((int)in==60) minute = 0;
     else minute = (int)in;
  }
  else{
    in= minute;
  }
};
```

```
  thing["water-time (1-30 mins)"] << [](pson& in){
      if ((int)in>0 && (int)in<=30){
        watertime = (int)in;
      }else{
        in=watertime;
      }
  };
  thing["umbraltierra"] << [](pson& in){
      if ((int)in>0 && (int)in<=25){
        umbraltierra = (int)in;
      }else{
        in=umbraltierra;
      }
  };
  thing["rain"] >> [](pson& out){
      if (isRain == 1)out=false;
      else out = true;
  };
  thing["valvula"] >> [](pson& out){
      if (aux == 1){
        out=true;
      }
      else{
        out = false;
      }
  };

  dht.begin();
  timeClient.begin();
}
```

```
void loop() {
  thing.handle();
  timeClient.update();
  if (XBee.available() > 21) {
  if (XBee.read() == 0x7E){ //byte de inicio
    for (int i = 0; i<20; i++){
      byte descartar = XBee.read(); // descarta todos los bytes
hasta llegar al de lectura de canal analogico
    }
    int analogMSB = XBee.read();
    int analogLSB = XBee.read();
    tierra = analogMSB*256 + (analogLSB & 0xF0)*16+(analogLSB &
0x0f);
    }
  }
  if (millis()> time1+5000){
   float h = dht.readHumidity();
   float t = dht.readTemperature();
   int luz = analogRead(LDR);
   if (isnan(h) || isnan(t)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
   }
   time1=millis();
  }//modo programación
    if (program == true && minute == timeClient.getMinutes() &&
hour==timeClient.getHours()+2 && auxClock==false &&
digitalRead(rain)!=0 && tierra>500){
      auxClock = true;
      digitalWrite(rele, HIGH);
      aux=true;
      auxminute = minute + watertime;
      if (auxminute > 59) auxminute = auxminute-60;
      thing.call_endpoint("email", thing["clima"]);
    }
```

```
    //modo automatico
    if (modo == true && tierra>1000-umbraltierra*10 &&
timeClient.getHours()+2<11 && timeClient.getHours()+2>18 &&
dht.readTemperature()<20 && analogRead(LDR)<400 &&
auxClock==false && digitalRead(rain)!=0){
      auxClock = true;
      digitalWrite(rele, HIGH);
      aux=true;
      auxminute = timeClient.getMinutes() + watertime;
      if (auxminute > 59) auxminute = auxminute-60;
      thing.call_endpoint("autoemail", thing["clima"]);
    }
    if((modo == true || program == true) && auxminute ==
timeClient.getMinutes() && auxClock==true){
      auxClock = false;
      aux = false;
      digitalWrite(rele, LOW);
    }
    if (aux==true && auxminute == timeClient.getMinutes()){
        aux = false;
        digitalWrite(rele, LOW);
    }
    if (digitalRead(rain)!=isRain){
        isRain = digitalRead(rain);
        thing.stream(thing["rain"]);  //actualiza lluvia
    }
    if (aux!=previous){
        previous = aux;
        thing.stream(thing["valvula"]); //actualiza valvula
    }
}
```